Flow/Interface Association for Multi-interface Mobile Terminals: e-Health case proposal

Mohamed Abdelkrim Senouci, Abdelhamid Mellouk
Université Paris-Est, LISSI, UPEC, 94400 Vitry sur Seine, France

***Abstract*—A heterogeneous wireless environment is composed of different networks technologies (e.g., LTE, UMTS, and Wi-Fi) each with different coverage and different capacity to cater for diverse service requirements. In such an environment, multiple access networks could be available for a mobile user. The latter can run multiple applications each has its particular Quality of Service (QoS) needs. Hence, it would be advantageous for multi-interface terminals to simultaneously use multiple network interfaces instead of switching from one interface to another to gain in performance. This involves the association of each flow with its suitable network that meets its specific requirements in a way that best maximizes the global terminal utility. Nevertheless, using multiple networks simultaneously may consume more energy than using only one interface. Therefore, the energy consumption should be considered as a criterion in the flow/interface association (FIA). This paper presents a novel method that takes into account network conditions, the monetary cost of the network, QoS requirements of the application, user preferences, and energy consumption of the mobile device to select the optimal FIA which achieves the best trade-off among all the considered criteria. We developed a real health monitoring testbed comprising multiple sensors for medical application to quantify the real benefit of our approach. The obtained results are supported by comparison with other works.**

***Index Terms*—Heterogeneous Wireless Networks, Network Interface Selection, Flow/Interface Association, Stochastic Optimization, e-Health.**

I. Introduction

The impressive development of wireless communication technologies has led to the emerging of HWNs, where the coexistence of different networks has become common in one area. Meanwhile, the evolution of mobile terminals equipped with multiple radio interfaces enable the mobile end-users to access the Internet through multiple network technologies. This heterogeneous environment has increased the opportunities for terminals with multiple interfaces (e.g., cell phones) to be always best connected, where the mobile terminals rank the available networks according to multiple criteria (service requirements, user preferences, terminal capacities, and network conditions) and select the network that best meets their needs anywhere at any time. However, in an overlapping area, the multi-interface terminal has the possibility to use two or more network interfaces simultaneously and not simply switch from one interface to another. The mobile terminals are different in capacities and can run simultaneously different types of applications over the Internet (e.g., interactive video gaming, streaming, VoIP, etc.) which leads to diverse flows with different exigencies in terms of quality of service (QoS) such as delay, jitter, packet loss rate, and throughput. These flows may need different kinds of communication technology. Therefore, it would be advantageous for the multi-interface terminals to associate each data flow with a network that meets its specific requirements. Associating each flow with a suitable interface is considered as a particular case of network interface selection [1], [2], [3] and is called flow/interface association (FIA) [4] throughout this paper.

In this multi-technology and multi-application environment, the FIA is a key, where the main target is to satisfy the individual flow's requirements while maximizing the global system performance; hence the association solution that provides the maximum total utility will be selected. The FIA problem involves the selection of which flow for which interface. The FIA is an optimization problem, and more specifically it is related to stochastic optimization problems. The latter primarily uses random search techniques where search depends on random variables. In [5], a meta-heuristic approach based on Tabu search (TS) has been proposed. Using simulation, the authors exhibit the standard algorithm poor performance to find the global optimal solution. Further analysis shows that the algorithm repeatedly gets entrapped in the local optima which prevents it from exploring other solutions. The authors identify the random diversification used in the standard Tabu implementation as the source of the problem, and a new method named "Oriented diversification" is proposed to remedy the issue. To achieve that, the method looks up the Tabu list for an application that has not been tested on one of the available networks and forces that selection in the newly generated association, thus preventing the diversification step from producing an association that is previously explored by the algorithm and overcome the entrapment issue. Comparisons through simulation show improvement in terms of results and computational time in favor of the proposed algorithm.

In this paper, we tackle the FIA problem where a multi-interface terminal running several applications with different QoS requirements tries to associate each particular application flow with its suitable network while maximizing the global terminal utility. Our objective is to select the association that best fulfills all criteria requirements of the flows at the least possible cost and efficient use of energy. To demonstrate the efficiency of the proposed approach, we developed a real health monitoring testbed compromising multiple sensors for medical application.

II. Our proposal

We chose TS [4] as the main algorithm to treat the problem of FIA for two reasons: (i) TS does not seize exploration unlike Simulated annealing (SA) whose exploration relies on the temperature variable which eventually becomes low and the algorithm remains steady, and (ii) TS can employ different types of memory (short, intermediate, and long-term) to bias move towards promising areas of the search space or promote a general diversity. To enhance the performance of TS, we propose three improvements to the original Tabu search as follows.

A. Clever Diversification

In clever diversification (CF), the search space from which the candidate is chosen is kept smaller, although it still grows with S but with a lower rate. This will increase the chance to stumble upon a neighborhood with global optimum. Second, the candidates produced by CF will contain interfaces that are likely to be part of the global optimum.

To explain how CF works, let us assume that we have four flows and three interfaces. The procedure to generate a candidate includes two steps:

1) Limit the search space, using the pseudo-code as defined in Algorithm 1. The algorithm's output is a matrix M, where each column represents a flow and the lines are the available interfaces for that flow.
e.g.
$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ - & 2 & 2 & - \\ - & 3 & - & - \end{bmatrix}$$
means flow 1 can use only interface 1, and flow 2 can choose interface 1, 2 or 3 and so on.
2) Generate a candidate: for each flow (f) we randomly assign an interface (i) as defined in M. If the resulted candidate is tabu, we repeat the process for a maximum of n times ($n > 0$). If no candidate can be generated then resort to random diversification.

B. Approximate Initial Solution

Tabu search generates a random candidate for an initial solution and starts improving upon it, sometimes the provided solution is too far from the global optimum and/or invalid and the algorithm will spend a considerable amount of time to improve it. Instead, we supply the algorithm an approximate solution with high utility and valid (if available), so the algorithm has a better chance to find the global optimum.

To obtain an approximate solution for each flow, we locate the interface with the highest utility and assign it to that flow, then update the interface's consumable resource (e.g., bandwidth, buffer, etc.) to reflect the allocating. Although it is a simple technique, it does improve the performance dramatically as we will demonstrate it in the next section.

Algorithm 1 Limit Diversification Space

```
Require: M empty matrix
 1: for $f$ in Flows do
 2:     for $i$ in Interfaces do
 3:         Compute the utility $U_{fi}$ using equations [2]:
            for upward parameters (e.g., throughput):
                $$f(x) = L - (L - b)e^{(-k(x-a))} \quad (1)$$
            $k$ is computed using the following equation:
                $$k = -\ln(1-p)/(\textit{Target point} - a),\ 0 < p < 1 \quad (2)$$
            for downward parameters (e.g., Jitter)
                $$f(x) = L - e^{(k-(x-a))} \quad (3)$$
            for monetary cost:
                $$\begin{cases} f(x) = 1 - x/u,\ x \geq u \\ f(x) = 0,\ x > u \end{cases} \quad (4)$$
 4:     end for
 5:     Compute the mean $m$ and std. deviation $sd$ of $U_{fi}$
 6:     $SD_{only}$ = False
 7:     for $i$ in Interfaces do
 8:         if $\neg SD_{only}$ then
 9:             if $U_{fi} \geq m + sd$ then
10:                 Clear $M[f]$
11:                 $SD_{only}$ = True
12:                 Add $i$ to $M[f]$
13:             else if $m - sd \geq U_{fi} \geq m + sd$ then
14:                 Add $i$ to $M[f]$
15:             else
16:                 Discard $i$
17:             end if
18:         else if $U_{fi} \geq m + sd$ then
19:             Add $i$ to $M[f]$
20:         end if
21:     end for
22: end for
```

C. Oscillation

By default, TS only accepts solutions that are equal or have better utility, this behavior renders the algorithm greedy and it might get stuck on local optima. This problem can be solved by allowing TS to accept and explore sub-optimal solutions. To achieve that, we use a mechanism similar to SA's acceptance probability (Equation 5), but unlike SA the probability here increases when the number of iterations since the last improvement gets larger (resp. c is set low). This will result in TS to be more stochastic when improvements ceased. Once a better solution is found, then the probability is reset to allow the algorithm to explore the vicinity of the current solution.

$$Oscillation_{Probability} + \frac{Last_{improvement}}{c} < random(0,1) \quad (5)$$

where c is a constant, $c \geq 1$.

In the following section, we will present the work carried out toward the design, development, and implementation of our testbed. We will also report the obtained results, supported by comparison with other works.

III. Design and Implementation of Real Health Monitoring Testbed

In this section, we present the implementation details of our proposition in an appropriately configured *RaspNet* and discuss the scenarios we have tested. More specifically, an e-Health application is considered to assess the selection algorithm impact on various parameters. Five medical sensors from Libelium [6] are used to capture different biometrics, namely ECG, EMG, GSR, Airflow and a Thermometer. The sensors are wired to an e-Health Sensor Shield(PCB) [7] from the same company which in turns connect to Arduino, Intel Galileo or Raspberry Pi, the latter is used in this study (see Figure 1). This setup (sensor node) can be used to monitor the state of a patient in real time or to get sensitive data in order to be subsequently analyzed for medical diagnosis.

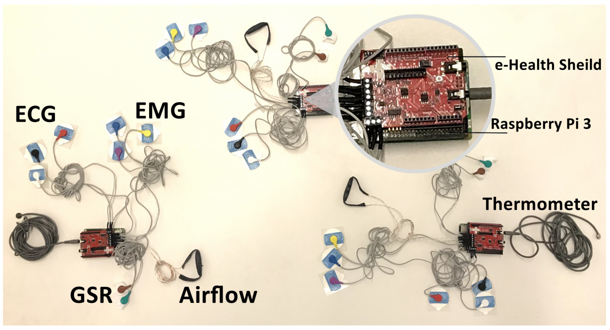

Fig. 1: The e-Health platform.

The gathered information is wirelessly sent over LAN using any of the 2 connectivity options available: Wi-Fi and Bluetooth to another Raspberry Pi (router node) connected to the internet. The router node has 5 wireless network interfaces, built-in Wi-Fi and Bluetooth for LAN communications with the sensor node; 4G, 3G and Wi-Fi dongle with internet access for data forwarding. The router node main objective is routing the received data from the sensor node or nodes to the server through the appropriate interface according to the application and user requirements using a selection algorithm (see Figure 2).

This entity takes into consideration several pieces of information, such as the monetary cost, the network performance, etc. Some of these characteristics are collected from the operating system of the device or are hard-coded due to lack of appropriate equipment.

The testbed is based on client/server architecture, both of which run on different Linux flavors.

A. Client

The client comprises two entities with different tasks. a **sensor node** that captures patient's vital signs using different

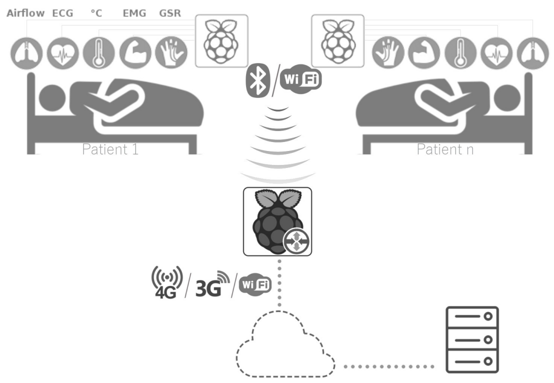

Fig. 2: The testbed setup.

medical sensors, and a second entity, known as **router node** responsible for routing the captured data via the most suitable network interface.

1) Sensor node: Runs on Raspberry pi 3 with raspbian stretch OS. It contains an e-Health Sensor Shield (PCB) mounted on top of the Raspberry pi 3, and an ECG, EMG, GSR, Airflow and a Thermometer connected to the e-Health Sensor Shield. Each sensor stores the captured data in a file with real-time updates, then the files are sent over Wi-Fi or Bluetooth in LAN setup to the router node.

2) Router node: Also runs on Raspberry pi 3 with raspbian stretch OS, it has access to the internet through 3 different radio access technologies 4G, 3G, and Wi-Fi. The node's main objective is sending the received data from other sensor nodes using the most suitable network interface according to the flow requirements and user preferences. The implemented system powering the node consists of five separate but complementary modules. The modules objectives are as follows:

1) **Network Probing** this module collects all the information about the parameters pertaining to network selection, such as the monetary cost, the network performance, the power consumption, the signal strength, speed, GPS position etc. the type of information collected depends on the application requirements. In this study, the first 3 information are considered.
   - **Power Consumption**: is difficult to measure accurately. An estimation could be made based on the wireless Network Interface Card (NIC) transmission power and data rate but this method does not consider the power consumption when the interface is idle or receiving data. Another approach would be to compute the terminal's power consumption over a period of time t (in seconds) with all network interfaces turned off, then doing the same with one interface turned on and operational (sending and receiving). The difference between these two measurements divided by t is the average network interface's power consumption per second. We adopted the second approach, and cal-

culated power consumption for all available wireless NICs, results are depicted in Table I.

- **Monetary Cost**: since it is fluctuant and based on providers' economic policies, the values for each interface are precomputed based on the carrier's data plan and hard-coded.
- **Network Performance**: refers to the measurement of the overall performance of a service, as seen by the customer. There are many different ways to measure the performance of a network, the following measures are often considered important:
  - **Throughput** is the actual rate that information is transferred.
  - **Latency** the delay between the sender and the receiver.
  - **Jitter** variation in packet delay at the receiver.
  - **Packet Loss** the number of corrupted packets expressed as a percentage or fraction of the total sent.

There are many tools available to measure one or more of these parameters, some of which are proprietary and specific to vendor applications. The measurement process is typically undertaken by sending a number of probing packets from one system (sender) to another (receiver), when the packets arrive at the receiver, some predefined metrics are computed and reported.

A more accurate method is to use dedicated software such as Spirent Test Center, JDSU QT600, Netcps, IxChariot, Iperf3, Ttcp, netperf, NetPIPE, Flowgrind or bwping for measuring different parameters.

We have chosen Iperf3 [8] a widely used tool for network performance measurement and tuning. It is open source and cross-platform that can produce standardized performance measurements for any network. Iperf3 has client and server functionality, and can create data streams to measure different parameters between the two ends in one or both directions. The data streams can be either TCP, UDP or SCTP. *Throughput* is measured with TCP while UDP is used for *Jitter* and *Packet loss*.

Iperf3 does not measure Latency, therefore another tool is required for that. Latency can be either One-Way Delay (OWD), i.e. latency in one direction or in terms of Round-Trip-Time (RTT), is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received. OWD is more relevant measure than RTT for applications that are primarily producers or consumers of data, since delays between two network nodes are not completely symmetric.

OWAMP [9] is an implementation of the One-Way Active Measurement Protocol ***OWAMP*** as defined by RFC4656. It is used to determine OWD. However, OWAMP drawback lies in its difficulty to use, it requires Network Time Protocol (NTP) synched client and server with at least 4 peers for the measurements to be accurate. Another method that is easy to accomplish (with simple ping) but would produce an approximation is to halve the RTT value, the accuracy of such an estimate depends on the delay distribution in both directions: as delays in both directions become more symmetric, the accuracy increases.

The tests are executed in parallel for all available NICs to reduce execution time, but that requires the server's throughput to be greater than all the client NICs' combined for the measurement to be as accurate as possible, otherwise, sequential execution is performed. Our server has enough throughput for a parallel approach. The execution time takes about 6 seconds in total, 3 to determine throughput and another 3 for the remaining parameters.

These tests can throttle the transmission of other data through the internet connection as they are undertaken, and can cause inflated data charges. Therefore, it stands to reason to run it sparingly or when needed. In this testbed, the module is executed routinely and asynchronously at a predefined interval, but the frequency is subject to change depending on different factors such as mobility, terminal speed, etc. It is also possible to force its execution for certain events, in this study that would be:

- Disconnected network.
- NIC is down or up

QoS, monetary and power consumption values of the used networks are detailed in Table I. The ones marked as dynamic are for reference only, since they are subject to change and needs to be computed during the program execution. Others are pre-computed and hard coded.

2) **Policies** they provide information about flow requirements and user preferences to the network selection module. Every flow has to define the required QoS and user preferences values. In order to simplify this issue for the less experienced users, we predefined a list of the most usable values as follows.
   - **QoS**
     - Real-time voice (e.g. PCM VoIP)
     - Real-time data (e.g. TelePresence)
     - Best effort voice (e.g. audio streaming)
     - Best effort data (e.g. video streaming)
     - Scavenger (e.g. FTP, P2P or Torrent)
   - **User preferences**
     - Performance
     - Cost
     - Energy
     - Balanced
3) **Network selection** The module is responsible for assigning each data flow to an appropriate network interface while considering multiple factors. It takes as input flows requirements, user preferences, and network interfaces measurements. The output is a priority list of the two most suitable networks for each flow as determined by the selection algorithm. Different algorithms will produce different results, but the main goal is to determine the

global optimum or at least approach it in case of large search spaces. Five selection algorithm are considered namely, Brute force, Smart TABU (proposed algorithm), TABU, Simulated annealing, Random. Brute force is used as a standard by which other algorithms are measured since it always finds the optimum solution. While the random strategy will simply assign each flow to a network interface randomly, and is meant to exhibit the efficacy of other selection algorithms.

4) **Transmission** It is implemented as threads, a thread for each data flow. The module objective is to send the data read from a file through the given network interface using TCP socket, if the network is disconnected or down then the second interface is used. If all interfaces are down the thread status is updated to "disconnected" and communicated to the main program to force a new network probing, followed by an interface selection. The selection results are communicated back to the thread, and transmission is resumed. The module will run continuously until all data are sent or discarded by the user (e.g. file deleted). The end of data flow is determined by setting a timer on the given file, if the timer runs out and no write event is reported by the inotify API then the flow is considered close, and the thread will exit.
5) **Main** This module implements the main execution routine and brings all other modules together. It monitors a given Folders for file creation events using Linux inotify API. In case of an event, a *Transmission* thread is created and assigned to the newly created file, then networks parameters and flow's information are fetched through the *Network probing* and *Policies* modules respectively. The network selection is executed once the previous information is available, and the results are fed to the Transmission thread to start sending the data. It also routinely monitors network events such as newly discovered networks or network disconnection, and reacts by forcing the Network probing execution followed by network selection, the new results are propagated to the active *Transmission* threads.

TABLE I: QoS, monetary and power consumption values of the used networks

2*Technology	2*Carrier	Dynamic				Static	
		Throughput (Mbits/s)	Latency (mSec)	Jitter (mSec)	PER (%)	Cost (cents/Mbyte)	Power consumption (mWatt)
4G	Bouygues	12	83	10	0	0.2	1726
3G	SFR	3.5	345	17	0	1	1678
Wi-Fi	Lab WiFi	5	65	23	1	0.001	210

B. Server

Uses Linux Ubuntu 16.04.4. The implementation is fairly simple, it accepts connection from clients and stores the received files locally, if a file already exists the server responds with the end-of-file position to the client. This is useful in case of disconnection so the client can resume sending from the last successfully sent packet. Another objective is to provide iperf3 server for network testing, it does so by running iperf3 server on multiple ports if possible for concurrent testing. The number of port is limited by the server's connection uplink and/or downlink capacity, depending on the test direction, the connection's throughput is required to be at least equal to the clients' connections throughput combined for the measurement to be accurate. In this case study, the server is running on Amazon EC2 with 70 Mbits/sec downlink and 65 Mbits/sec uplink internet connection, which is more than enough to provide concurrent testing for one client with 3 internet connections rating at 20 Mbits/sec max.

IV. Results

In this section, we will discuss the testbed setup, execution and the obtained results. Real traffic through the Internet is used, with the clients located at Paris, France, and the server at Ohio, United States. The following process was followed in order to setup the testbed before each test case:

1) The captured data from each sensor is stored in a file, and the same file is used for all subsequent tests, in order to prevent data size changes between executions.
2) The router node is configured in an initial state where the sensors' files are already in local storage. Thus isolating any variations introduced by wireless communications between the sensor and router nodes.
3) The router node is not connected to any wireless access network and no information is transferred.
4) The flow requirement is parameterized according to Table II. While the user preference is set to ***Cost*** for all traffic.
5) The network interfaces are connected and the scenario is executed.

TABLE II: Sensors' network requirements

Sensor	File size (Mbyte)	Requirement
Airflow	1	Real-time data
Thermometer	4.5	Real-time data
EMG	3.5	Best effort data
GSR	2	Best effort data
ECG	55	Scavenger

The execution starts with the router node initiating a network probing and collects all the information about the network's parameters, then running one of the network selection algorithms. Selection algorithm except Brute force can run indefinitely, therefore a stopping condition is needed to terminate the algorithm's execution. We limited the running time to 20 seconds or when the results seize improving for 100 consecutive iterations.

Once an association is obtained, each file is sent through the assigned network. Results are acquired after all data are transferred successfully. Network disruptions during execution will affect the obtained results and, as such, are discarded.

Results are described by satisfaction ratio which represents to how extent the user and application (flows) requirements are met. 1.0 means all the requirements are satisfied, while 0.0 is the antithesis.

For each selection algorithm, there are three different outputs representing the number of sensor nodes used in the test, 1, 2 and 3 nodes which yield 5,10 and 15 flows respectively. Each test case is repeated ten times and results are averaged. Output is depicted in Figure 3.

Figure 3. shows the satisfaction ratio. Using 5 flows, all selection algorithms produced SR equals to 1, except the random solution. By increasing the number of flows the SR starts to drop reaching its lowest value 0.67 with the random solution while TABU and simulated annealing perform slightly better at 0.75 and 0.71 respectively. Smart TABU (STABU) gives results identical to the Brute force algorithm.

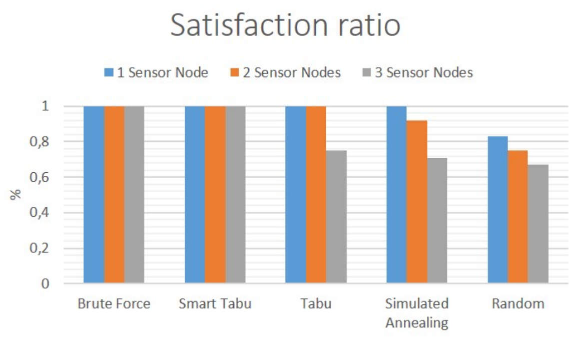

Fig. 3: Satisfaction ratio using different selection algorithms while increasing sensor nodes.

In order to improve the performance of STABU and enhance the end-user quality of experience (QoE), we will consider in our testbed the feedback received from other users that have already interacted with that a specific network operator. A network reputation factor will be then computed based on the feedback received from other users and the user past interaction with that network as well, which can guide the FIA process. Of course a credibility factor will have to be considered for the users' feedback.

V. Conclusion

In this paper, we tackled the FIA problem in heterogeneous wireless networks. The FIA can be considered as a particular case of network interface selection, where a multi-interface terminal running multiple application can associate simultaneously each application flow to its suitable interface. Associating each flow to an interface that meets its requirements in a way that maximizes the global system performance is an optimization problem, and more specifically it is related to stochastic optimization problems. To solve this problem, we proposed a new approach named Smart Tabu Search. In order to assess the effectiveness and efficiency of the new approach, we developed a real health monitoring testbed comprising multiple sensors for medical application. The obtained results clearly show the superiority of the new approach.

References

[1] M. A. Senouci, M. S. Mushtaq, S. Hoceini, and A. Mellouk. Topsis-based dynamic approach for mobile network interface selection. *Computer Networks*, 2016.

[2] M. A. Senouci, S. Hoceini, and A. Mellouk. Utility function-based topsis for network interface selection in heterogeneous wireless networks. In *Proceedings of IEEE ICC*, pages 1–6, Kuala Lumpur, Malaysia, 2016.

[3] M. A. Senouci, M. R. Senouci, S. Hoceini, and A. Mellouk. An evidential approach for network interface selection in heterogeneous wireless networks. In *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, Washington, DC, USA, 2016.

[4] M. A. Senouci, H. Senouci, S. Hoceini, and A. Mellouk. Flow/interface association for multihomed mobile terminals in heterogeneous wireless networks. In *Proceedings of IEEE International Conference on Smart Communications in Network Technologies (SaCoNeT)*, pages 171–176, El Oued City, Algeria, 2018.

[5] F. H. Mirani, N. Boukhatem, and P. N. Tran. On terminal utility for multiple flow/interface association in mobile terminals. In *2011 IFIP Wireless Days (WD)*, page 1–6, Niagara Falls, Ontario, Canada, 2011.

[6] Mysignals. https://www.cooking-hacks.com/mysignals-sw-ehealth-medical-biometric-complete-kit.

[7] Connection bridge. https://www.cooking-hacks.com/raspberry-pi-to-arduino-shield-connection-bridge.

[8] Iperf3. https://iperf.fr/. Iperf3.

[9] Owamp. http://software.internet2.edu/owamp/.