DEVELOPING DEEP SIMULATION SYSTEMS TO STUDY WIRELESS NETWORK PERFORMANCE

Paolo Pileggi[1], Giuseppe Iazeolla[2] and Pieter Kritzinger[3]

[1]Networking Group, University of Roma 'Tor Vergata', Italy
[2]Software Engineering and System Performance Modelling Group,
University of Roma 'Tor Vergata', Italy
[3]Communications Research Group, University of Cape Town, South Africa

psk@cs.uct.ac.za(Pieter Kritzinger)

Abstract

There is an abundance of literature that present simulation studies of wireless networks, including IEEE 802.16 or WiMAX. The focus in such papers is invariably on the results and never on the simulation itself, which is seen merely as a means to determine numerical values. The reason for this may be that, with very few exceptions, the simulations use one or other well-known simulation platform. In this paper we argue that, with the complexity of a system such as WiMAX, one needs to develop a simulator from scratch by implementing the basic simulator components using a programming language, such as Java. This ensures that every detail of the standard is understood and represented and that all system parameters and their values are accessible to the modeller. Moreover, the performance metrics one wants to access may not necessarily be those offered by the simulation platform nor is it always clear whether the statistical methods follow those that the user has confidence in. This paper does not deal with the performance of a wireless network per se but instead explains a general simulation methodology for determining the QoS and performance management of wireless networks that support differentiated traffic services.

Keywords: Simulation, Wireless systems, IEEE 802.16, WiMAX, system performance model, network of queues.

Presenting Author's Biography

Paolo Pileggi is a PhD student at the University of Rome II 'Tor Vergata', Italy. He obtained his Bachelors, Honours and Masters degrees with distinction in Computer Science from the University of Cape Town, South Africa. Currently, his work focuses on performance of wireless networks, particularly 4G technology.

1 Introduction

The research literature abounds with work on the performance simulation of wireless systems, including IEEE 802.11 and IEEE 802.16 [1, 2, 3, 4, 5], all reporting on simulations to find various system performance indices such as throughput, latency times, jitter, etc. In all these papers, simulation is only seen as a means to obtaining the numerical predictions. In this paper we do not present another such model for performance prediction. The paper is focused, instead, on the presentation of a general methodology that illustrates the gradual steps of analysing, specifying and developing a simulator of a wireless network, using IEEE 802.16 or WiMAX as a concrete example.

For the sake of conciseness and following the example of the authors in [6, 7, 8], we assume the reader is familiar with the WiMAX standard and with the details of the physical (PHY) and Medium Access Contention (MAC) layers, in particular the WirelessMAN-SC^{TM} PHY [9]. We consider a fixed WiMAX network in a point-to-multipoint (PMP) topology in particular. Under the PMP topology, Subscriber Stations (SSs) do not communicate directly with one another but have to do so via the Base Station (BS). A further detail that is important to know for what follows is that connections are granted per SS (GPSS) by the BS, rather than the alternative of granting requests per-connection. Fig. 1 shows the topology of the system of our application example: Local area networks ($ENV_1, \ldots, ENV_N$) are connected via their respective SS to a single BS which acts as a back haul network relay to other SSs on a wireless link or to the Internet (INET) via a fixed-line modem.

The *fixed* WiMAX standard specifies four different categories of traffic (called *Traffic Categories*, TCs): Unsolicited Grant Service (UGS), real time Polling Service (rtPS), non-real time Polling Service (nrtPS), and Best Effort (BE). Resource and connection management is an important aspect of the IEEE 802.16 back haul network because it controls the resources between the ENVs and the Internet. It therefore has a significant impact on the QoS. The BS manages QoS by sharing resources between the multimedia connections requested by the users, which originate in the ENVs and the Internet. The network must enforce connection admission control (CAC) and scheduling protocols to maintain acceptable QoS for admitted connections. These connections carry typical Internet data, such as hypertext transfer protocol and Voice over Internet Protocol data, which subscribe to one of the TCs specified in the IEEE 802.16 standard. Only once a connection has been granted service, may it start generating and sending data to its destination. Each datum of a granted connection is associated with the TC that the connection subscribed to and receives service accordingly.

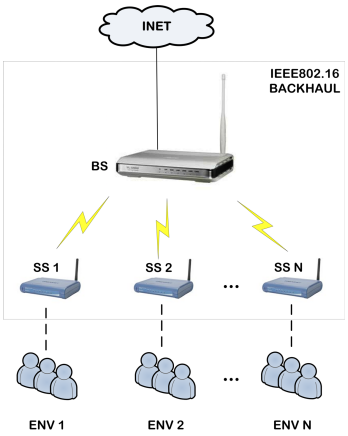

Fig. 1 The vertically integrated wireless Internet scenario with the IEEE 802.16 back haul network

In what follows, we illustrate a methodology to develop a discrete-event simulator ensuring that the implementation follows the detail of the standard, that the various parameter values used are known, while obtaining reliable simulation result in reasonable times with standard computer equipment. The paper is organised as follows: Sect. 2 gives an overview of our simulation approach. Sect. 3 describes the simulated system. Sect. 4, before the conclusion, gives the details of the simulator development.

2 Simulation Approach

For a simulation, one can either apply existing platforms, such as the OPNET Modeler [10], ns2 [11], QualNet [12] or OMNET++[13], or develop one's own simulator by implementing the simulation engine in a language such as Java. The OPNET Modeler Wireless Suite, as one instance, provides high fidelity modelling, simulation, and analysis of a broad range of wireless networks. But these simulators are complex, general-purpose software suites and it is seldom clear which details of the network stack are being modelled or where all the associated parameters may be found. More controversially, some of these systems are commercial products and, for proprietary reasons or otherwise, do not make clear [14] whether

- one is assured that the simulation has stabilised before sampling,
- what the sample sizes are, or
- whether sampling is done to ensure identically, independently distributed (i.i.d.) variables.

In a controversial paper, Cavin [15] illustrated these points by showing the deviation in results among widely-adopted simulators, such as OMNeT++ and ns2, for a sample mobile ad-hoc network experiment. Statements such as *" . . . users are responsible for verifying for themselves that their simulations are not invalidated because the model implemented in the simulator is not the model that they were expecting . . . "* [11] are not very encouraging when one is conducting a scientific study. Nevertheless, these general simulation platforms and libraries are apparently useful as witnessed by the many published studies that use them.

Some authors, such as Bianchi *et al.* [1] and the authors [5] prefer to know exactly what is being simulated and therefore prefer the second option. They

- sacrifice the convenience of these software platforms and, at the same time,
- spend much time delving down into the minute system detail to ensure that these are represented by the simulation model.

It is then possible to measure and record exactly those performance values relevant to the study. In contrast to the use of general purpose simulation platforms, we call this *deep simulation*. Deep simulators make no claim to be general or to replace the existing general simulation platforms.

We used the general programming language Java, whose flexibility, extensive libraries and algorithmic capabilities supersedes the higher level languages provided by platforms such as OPNET Modeler, ns2, QualNet or OMNET++.

3 Conceptual System Development

The methodology we are about to describe has two main stages: We start with a system performance model (SPM) and from that extract the associated network of queues (NoQ) model to be implemented as a discrete-event simulator.

3.1 System Performance Model

The aim of the system performance model (SPM) is to identify and describe the network structure and management components chosen to be modelled. The SPM, shown in Fig. 2, presents a high-level abstraction of the network application. The abstracted functional frame, shown in Fig. 3, describes the TDD frame abstraction, i.e., the Downlink (DL) MAP (DLMAP), Uplink (UL) MAP (ULMAP), and so on, as described in the standard [9]. The functional periods, shown in this frame (which we modelled) are the MAPs, DL profiles, UL connection contention period and the individual SS UL profiles. This frame is stripped of all fixed preambles and the maintenance opportunities periods which we assume to be utilized sparingly.

The network carries bandwidth requests (BWRs) and data per connection. Data arrive at each individual SS, where they are queued at the SS connection queues. An SS transmits data from these queues during its scheduled period in the UL subframe to the BS. The BS, in turn, receives data according to the ULMAP it communicated at the beginning of the current frame.

Each SS employs its own scheduling algorithm since, as already mentioned, the network is operating in GPSS mode. The BS is therefore only responsible for scheduling each SS UL time and each SS must schedule its own per-connection data transmissions. Data are transmitted from an SS to the BS over the wireless UL-server (W_{UL}) and from the BS to an SS over the wireless DL-server (W_{DL}).

The fixed-line input/output physical links (F_{in}/F_{out}) at the BS concurrently accept data arrivals from, and transmits data to, the Internet (INET), respectively. Data is transmitted over servers F_{in} and F_{out} between the INET node and the BS at a very high data rate such that the time taken for data to be transmitted across F_{in} and F_{out} is close enough to zero that one may safely assume the instantaneous arrival of data over each link.

In the system, BWRs arrive at an SS and are buffered there until the connection contention period, specified in the ULMAP, starts. During this period, BWRs are transmitted to the BS according to a truncated exponential backoff process to help with collision avoidance and are queued at the BS CAC. In the model, BWR arrivals at the SS are forwarded directly to the BS CAC waiting-line and no collisions of BWRs are assumed. Also, all BWRs in the CAC waiting-line are evaluated at the end of the connection contention period. These assumptions are made since the detailed collision resolution process does not impact the performance of the system significantly.

As shown in Fig. 4, at the end of the connection contention period of the current frame F_n, the CAC processes BWRs and keeps the granted requests as connection set S_A. The CAC then informs the BS scheduler of its admission decisions during the current frame F_n. The BS scheduler does not use this information, i.e., set S_A, to calculate the new MAPs for the next frame F_{n+1}. Rather, the BS first informs the SSs during the frame F_{n+1} of the admitted connections in S_A and only uses the information in the grant request set S_A to calculate the MAP for frame F_{n+2}.

The BS scheduler uses virtual queue information, together with the scheduling algorithm and QoS parameters, to determine the new ULMAP and DLMAP. These MAPs are transmitted from the W_{DL} to the SSs at the beginning of the next frame.

Data per connection arriving at the BS are queued at the BS MAC memory buffers. The fixed-line output and wireless UL server (F_{out} and W_{UL}, respectively) each have their own associated logical waiting-line structures where, in the case of F_{out}, the service time is zero, as explained before. this server.

3.2 Network of Queues Model

In this section, we develop the network of queues (NoQ) model our simulator is based on. The model shows only one TC for ease of explanation. Fig. 5 shows the data sources and abstractions of the Internet UL and the UL resource sharing between the N SSs.

As shown on the left of Fig. 5, the INET source generates data, i.e., BWRs and data per connection (data), served by the fixed-link UL server F_{in} and, since the fixed-link UL is high-speed, F_{in} is assumed to enforce no delay on data. The data arrive at the BS along the fixed-link UL interface.

Also, on the left of Fig. 5, it is shown how SS sources generate data that are queued at the respective SS's connection data queues. The wireless UL server

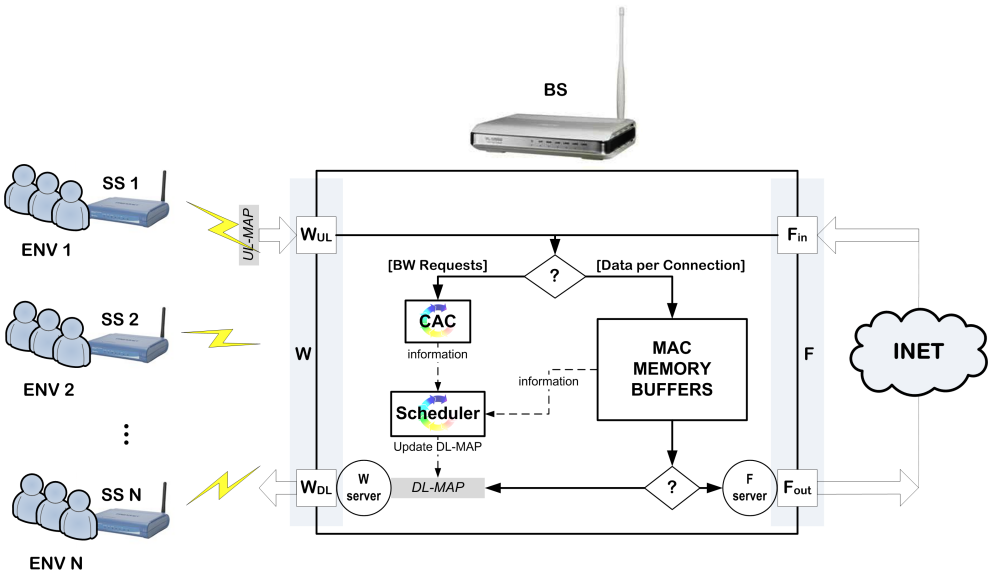

Fig. 2 The systems performance model

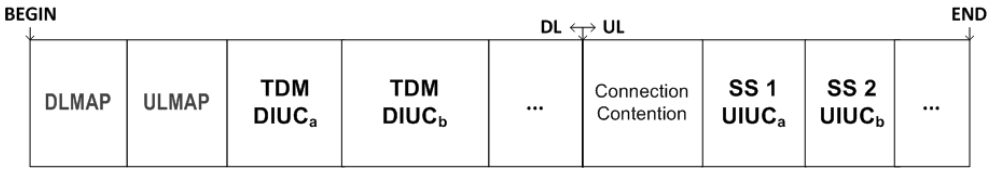

Fig. 3 TDD frame simplified by ignoring pre-ambles, periods necessary for antennas to switch between receive and transmit modes and other management time durations

W_{UL} serves the SS connection queues according to the ULMAP specified by the BS, where each SS executes the SS scheduler and serves its internally managed connection queues during the UL period allocated to it. Each SS transmits data along the wireless UL at a certain rate (specified in the information elements of the ULMAP) and data arrive at the BS along the wireless UL interface.

Data per connection arrive at the BS wireless receiver from the SS NoQ model, shown to the right of Fig. 5. Wireless UL data have as their destination either the fixed-line Internet or another wireless SS. Data units are therefore either queued at the MAC memory buffers for wireless SS-designated data or transmitted to the Internet along the F_{out} fixed-line link. Data arriving from the Internet at the BS along the F_{in} fixed-line interface have as ther destination some wireless SS and are thus queued at the MAC memory buffers.

The BS transmits data from the MAC memory buffers along the DL, as shown on the right hand side of Fig. 5. The BS scheduler is, as shown in the figure, responsible for generating the MAP information and transmitting it to the SSs.

BWRs are generated at both the SSs and Internet sources. As described in Sect. 3.1 and illustrated by Fig. 4, connections admitted and rejected during the current frame are first notified of admission or rejection during the DL of the next frame. Only after this period will the scheduler include these admitted connections in the MAP assembly decisions. The assumptions regarding BWR arrivals and evaluation have been described in Sect. 3.1 and hence we derive the NoQ model abstracting both wireless and fixed-line BWRs at the BS per TC, as shown at the top of Fig. 5. BWRs originating at the fixed-line and wireless sources are queued at the CAC. The processed requests are then forwarded to the BS scheduler.

Fig. 5 therefore shows the model that represents the entire system to be modelled based on the TDD frame structures and SPM described above.

3.3 Baseline Model

The baseline model is the system model definition excluding the resource and connection management components and simulation execution parameters. It includes the PHY layer, workload model, frame aspect ratio and fragmentation parameters. The baseline model was devised in order to ensure that the experimental results are that of a plausible system scenario bearing in mind that the current system is only used as an example to illustrate our methodology. As an example, a PHY layer parameter configuration is given by Table 1. Note that the maximum size of PDU fragments may effect the system performance: The BS allocates UL transmission time to each SS. However, an SS can

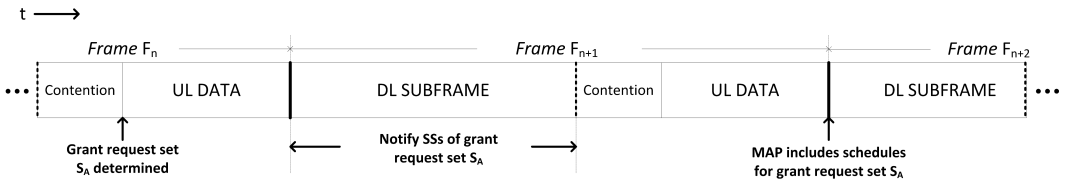

Fig. 4 Request admission-notification process

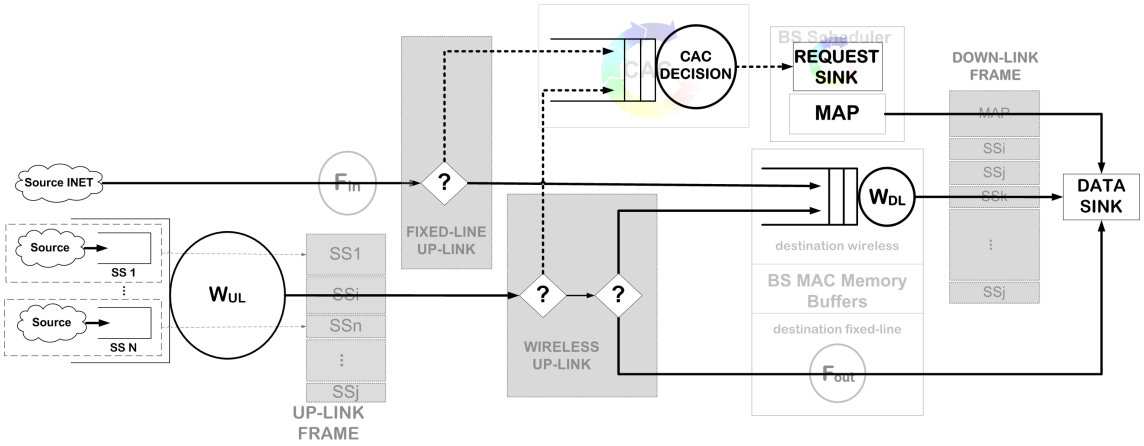

Fig. 5 Network of queues model model abstracting both wireless and fixed-line data and BWRs at the BS per TC

Parameter	Value
Frame duration (T_{frame})	$1ms$
UL wireless modulation rate (MCS_{UL})	80 Mbps
DL wireless modulation rate (MCS_{DL})	40 Mbps
Connection contention duration (T_{conn})	$50\mu s$
DL:UL frame ratio	668:332
Maximum PDU fragment size (C_{frag})	1280 bits

Tab. 1 Physical Layer parameter values selected for the baseline model

only transmit a PDU if there is sufficient time to transmit the entire PDU. If the SS is allocated time less than that of the fragment size it will not be able to transmit that PDU/fragment. If the BS is not able to allocate enough bandwidth during system operation to an SS, that SS becomes deadlocked.

We also know that all SSs are scheduled along the UL during each frame. Furthermore, in our system, all SSs are treated equally in terms of workload and scheduling process parameter values. Therefore we must determine the PDU fragment size C_{frag} (in bits), such that $0 < C_{frag} \leq C^S_{UL}$, where C^S_{UL} is the mean amount of data that each SS may be able to send per UL frame for some number S of SSs.

Considering the MAP transmission time T_{MAP} and connection-contention period T_{conn}, T_{UL} is given by Eq. 1. Furthermore (referring to (cf. Table 1), consider the choice of $T_{conn} = 50\mu s$, $\omega_{UL} = 0.332$ and $S = 6$ and where T_{MAP} is given by Eq. 2. With $C_{frag} = C^S_{UL}$, where $C^S_{UL} = \frac{T_{UL} \times MCS_{UL}}{S}$, the calculation shows that one would have to choose $C_{frag} \leq 3723$ bits. The length of a VoIP fragment is 1028 bits which is what we have chosen for C_{frag}.

$$T_{UL} = \omega_{UL} \times (T_{frame} - T_{MAP}) - T_{conn} \quad (1)$$

$$T_{MAP} = \frac{140 + 32 \times S}{MCS_{DL}} \quad (2)$$

Since the UL scheduler allocates UL transmission periods based on the current connection status of the system, it may be possible that an SS may experience a period during which it cannot transmit any data. However, since the connection-level behaviour is dynamic and SSs are considered the same, the SS will, in time, recover from this congestion state.

3.4 Execution Parameters

The simulation accepts a set of execution parameter at the start of an experiment. Table 2 gives an example of an execution parameter set. For the example parameter values provided in this paper, we separately calculated that, with a third of the radio frame dedicated to UL data transmission, the system is fully utilised at a rate of 4.4 Mbps [16]. We therefore selected a range of traffic intensities below 4.4 Mbps for our experiments. From preliminary runs, it was apparent that the initial transient was over at $t = 1000$s. We decided to run the simulations for 6500s because we wanted to obtain as much performance data as possible given the amount of available RAM and ROM memory. This provided us with enough simulation time from which we could obtain performance estimates. The initial seeding value of the random number generators is arbitrarily chosen and provided for repeatability of the experiments.

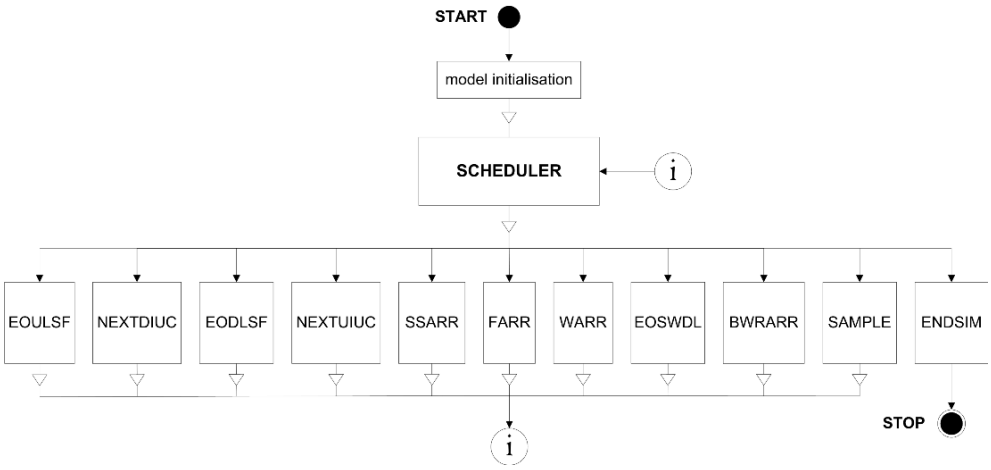

Fig. 6 Basic simulation engine PFC with the process flow blocks of the event routines

Parameter	Value
SS arrival rate (minimum)	1.6 Mbps
SS arrival rate (increment)	0.4 Mbps
SS arrival rate (maximum)	3.6 Mbps
Number of SSs	6
Simulation duration	6500s
Tracing starting time	0s
Random number generator seed	11111

Tab. 2 Experiment execution parameter values

4 Simulation Engine Design

The simulation engine is event-driven [14], where an event represents a discrete change to one or more state variables at a particular moment in time. The main components are the scheduler and event routines. The engine is organised as shown in Fig. 6. An event routine is invoked by the scheduler at the time that an event of the associated type occurs. As shown in Figure 6, 11 events are taken into consideration by the engine: 4 events were identified from the abstracted frame, as annotated in Fig. 7, and are

1. EOULSF: End of UL subframe.
2. NEXTDIUC: Next DL interval usage code, representing the start of the next DL profile.
3. EODLSF: End of DL subframe.
4. NEXTUIUC: Next UL interval usage code, representing the start of the next UL profile, including the UL connection contention period.

Using the NoQ model, previously described in Sect. 3.2, 5 events were identified, as annotated in Fig. 8. These are

1. SSARR: SS arrival, representing a PDU arrival at one of the SSs in the list of SSs.

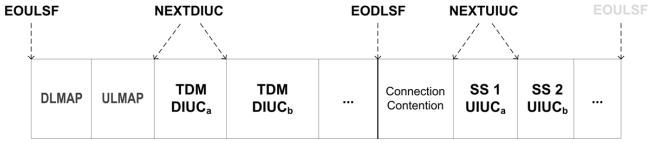

Fig. 7 TDD frame structure with relevant events annotated

2. FARR: Fixed-line arrival, representing a PDU arrival at the BS from the INET.
3. WARR: Arrival at the BS wireless interface, representing the end of service of a PDU along the UL.
4. EOSWDL: End of service of the wireless DL server.
5. BWRARR: BWR arrival, representing the arrival of an SS or Internet BWR at the BS CAC.

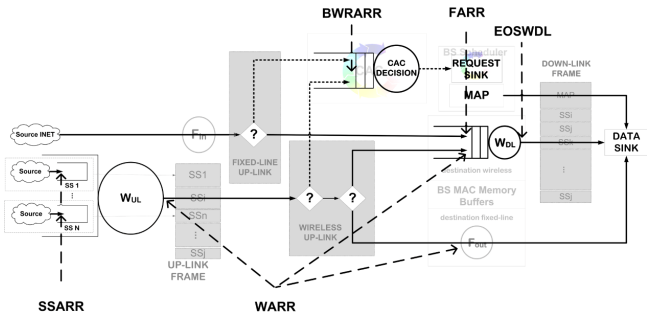

Fig. 8 Overall NoQ diagram with relevant events annotated

Finally, 2 events were identified to account for data sampling (SAMPLE event) and ending the simulator (ENDSIM event) respectively.

For each event, a detailed process flow chart (PFC) was produced and eventually translated into the simulation

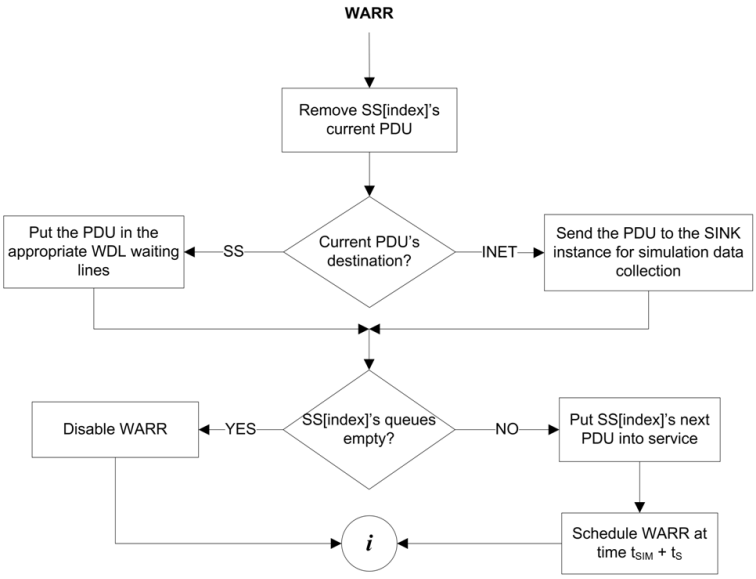

Fig. 9 Process flow chart for the WARR event

program. As an example, Fig. 9 shows the PFC for the WARR event: The current in-service PDU, at the SS that was last to transmit, is taken out of service and removed from the SS buffers. If the PDU's destination is the Internet, it is sent to the data sink for data collection. Otherwise, if it must be forwarded to another SS, it is put in the BS buffers, i.e., the WDL waiting-lines. Next, if the transmitting SS's queues are empty, WARR is disabled. Otherwise, the next PDU is put into service at the transmitting SS and WARR is scheduled at time t_{SIM} + t_S, where t_{SIM} is the current time in the simulator, i.e., the artificial simulation time, and t_S is the time taken to service the particular PDU being served.

As output, trace files are generated during simulation for both connection– and packet-level information. This allows one to use a sampling technique of one's choice, such as repeated runs or batch means methods [17]. Moreover, for such a complex simulation, it takes a long time to complete a simulation run and therefore, by generating trace files, methods to ensure statistical soundness, such as removing initial biased samples [14, 18, 17], can be redone without having to re-execute the entire simulation run.

Full details of the implementation, as opposed to the principles we have described, are clearly beyond the scope of this paper. The reader is referred to the MSc dissertation by Pileggi [16] for detailed documentation on the simulation engine, event process flows and interface specifications, as well as for detailed test case reports.

4.1 Testing the Simulator

The simulator was validated through the derivation of the various abstract models, carefully developed for the system under study. Our testing verifies that the simulator was built correctly, acknowledging that the simulator state space is simply too large to exhaustively test every function.

Due to the component nature of the simulator, a modular testing approach was taken: CAC, schedulers and workload generator components were tested first. Thereafter, the simulation engine was tested before the integration of these components was tested.

Positive testing was done to ensure the simulator components affect the system state variables in the way they should be affected. Negative tests, whereby it is tested whether the simulator *"... does not do what it is not supposed to do"*[19], were not conducted. Structural white-box tests were conducted to ensure that the simulator event process flows were correctly translated from PFCs into code: Integration testing, was done by inspecting the program source, i.e. tracing the method-chains invoked to the point where the required system state variables are affected.

5 System Capabilities with respect to the State of the Art

As seen above, the IEEE 802.16 technology and the vertically integrated wireless Internet scenario are particularly complex. Additionally, the study requires specific abstractions to be made. As said in Sect. 1, we therefore developed a deep simulator. It is generally accepted that simulation tools make simulation model development easier, faster and more effective and may include features such as experimental design, advanced reporting and exploitation of computer architecture to improve execution speed. Users of these tools may encounter some obstacles while learning to use them at first. Thereafter, simple simulation models may be developed fast and efficiently. However, developing simulation models of a more complex nature is a much more

challenging task; it requires users to have an expert understanding not only of the system but of the tool used as well.

By the mere fact that such tools need to be general, they present users with certain limitations, primarily due to the structure of the simulator in terms of the simulation paradigm followed and the language(s) used to realise the model. For simple models these tools are appropriate but for more complex ones, they are not. Users may want to (re-)configure the system to a larger extent and/or at a lower level, such as making changes to certain protocols fixed in the assumptions of the tool.

As one example, OMNeT++ is a modular general-purpose simulation environment with graphical and statistical components and offers a wide range of software suites, such as its mobility framework. However, it is seldom clear which details of the network stack are being modelled and where the associated parameters may be found.

For proprietary reasons, commercial products may not always make clear the particular techniques used for obtaining statistically significant [17] performance estimates. Furthermore, should the modeler require non-conventional data analysis, as was the case for this study, or simply inspect the performance data during data analysis, proprietary tools may not necessarily provide such functionality.

Most importantly, the accuracy of a simulation model developed, using a simulation tool, is also questionable. The study by Cavin et al. [15] we mentioned, shows how different implementations of the same simulation model, each developed using a different simulation tool, may result in a significant divergence between the results obtained from these simulators. This is partly due to the different levels of detail provided for implementation and configuration.

Simulation tools may prove very accurate if the abstraction-level and simulation paradigm supported suit the system being modelled. Otherwise developing a deep simulator may prove not only more accurate but also more useful by allowing the developer a greater degree of flexibility of manipulating the system model and allow the developer greater insight into the true nature of the system.

6 Conclusion

It is rare though to find performance study that uses a simulation, while providing a great deal of other information, and would mention much about the simulator. The reader is simply asked to believe that a simulator was developed, almost exclusively using existing platforms such as ns2, and that it is correct and the statistical methods employed are trustworthy.

The objective of this paper was not to disprove the validity of earlier work, although studies exist which put a question mark behind the use of simulation libraries, but rather to present a step-by-step methodology when a modelling team decides to build their own simulator from scratch in a process we call *deep simulation*.

While time consuming, the authors are of the opinion that knowing exactly what is being represented in the model, the facility to change the simulation, such as one needs to do when studying different schedulers and Connection Admission Control algorithms, and knowing the correctness of the underlying statistical methods employed, far outweigh the additional effort involved.

Acknowledgements

This work has been kindly supported by

- the University of Rome 'Tor Vergata'/CNIT (Italy),
- the FIRB projects on *Software frameworks and technologies for distributed simulation* and *Performance evaluation of complex systems*, by the University of Rome 'Tor Vergata' research on *Performance modeling of service-oriented architectures* and the CERTIA Research Center (Italy), and
- THRIP, TELKOM, Siemens-Nokia, Telesciences (South Africa).

7 References

[1] Giuseppe Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, March 2000.

[2] Chen-Nee Chuah. *A Scalable Framework for IP-Network Resource Provisioning Through Aggregation and Hierarchical Control*. PhD thesis, University of California at Berkeley, 2001.

[3] David Erman, Dragos Ilie, and Adrian Popescu. BitTorrent Session Characteristics and Models. In *Proceedings of the Third International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs'05)*, Ilkley, United Kingdom, 2005.

[4] Jiongkuan Hou, Jie Yang, and Symeon Papavassiliou. Integration of Pricing with Call Admission Control to Meet QoS Requirements in Cellular Networks. *IEEE Transactions on Parallel and Distributed Systems*, PDS-13(9):898–910, Sep 2002.

[5] Giuseppe Iazeolla, Pieter S Kritzinger, and Paolo P Pileggi. Modelling Quality of Service in IEEE 802.16 Networks. In *Software, Telecommunications and Computer Networks, 2008. SoftCOM 2008. 16th International Conference on*, pages 130–134, 2008.

[6] Jianfeng Chen, Wenhua Jiao, and Hongxi Wang. A service flow management strategy for IEEE 802.16 broadband wireless access systems in TDD mode. In *IEEE International Conference on Communications*, pages 3422–3426, 2005.

[7] Hyoung-Kee Choi and John O Limb. A Behavioral Model of Web Traffic. In *ICNP '99: Proceedings of the Seventh Annual International*

Conference on Network Protocols, page 327, Washington, DC, USA, 1999. IEEE Computer Society.

[8] I C Msadaa, F Kamoun, and F Filali. An Adaptive QoS Architecture for IEEE 802.16 Broadband Wireless Networks. In *Mobile Adhoc and Sensor Systems (MASS 2007). IEEE Internatonal Conference on*, pages 1–3, 2007.

[9] IEEE. *IEEE Standard for Local and Metropolitan Area Networks*. IEEE 802.16 Standard, 2004.

[10] O T Incorporated. OPNET Modeler. http://www.opnet.com.

[11] NS-2. The Network Simulator - NS-2. http://www.isi.edu/nsnam/ns.

[12] S N Technologies. Introducing the VisNet Network Planning Software. http://www.scalable-networks.com.

[13] OMNET++. A discrete event simulation system. http://www.omnetpp.org.

[14] Jerry Banks, John S Carson, Barry L Nelson, and David M Nicol. *Discrete-Event System Simulation*. Prentice-Hall, Third edition, 2000.

[15] David Cavin, Yoav Sasson, and André Schiper. On the accuracy of MANET simulators. In *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 38–43, New York, NY, USA, 2002. ACM.

[16] Paolo Pileggi. Cross-layer RaCM Design for Vertically Integrated Wireless Networks. Master's thesis, University of Cape Town, December 2009.

[17] Averill M Law and David W Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, 2000.

[18] Hisashi Kobayashi and Brian L Mark. *System Modeling and Analysis*. Prentice Hall, New Jersey, 07458, 2009.

[19] John Watkins. *Testing IT - An Off-the-Shelf Software Testing Process*. Cambridge University Press, 2001.