International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Automating Network Simulation Tool and Data Processing in Ad Hoc Networks

Halim Berradi, Ahmed Habbani, Mohammed Souidi, Hicham Amraoui, Nada Mouchfiq

Abstract*: One of the crucial and utilized simulator for mobile ad hoc network is the NS3 (Network Simulator). NS3 executes simulation scenarios and produces data regarding which flow monitor (flowMon) files are counted one of the important used format for estimating the research proposal. This article will introduce a novel framework to facilitate the simulation process in mobile ad hoc networks (MANETs). Generally speaking, when network researchers need to evaluate the experiments, there is generally the necessity to estimate the different models advantages by computing a set of QoS metrics to understand the network performance impact in such scenario. Network simulators, and NS3 specifically, need important program writing from the researcher to consolidate simulation results. This article states a contribution for NS3 composed of a new Framework that tends to make it less complicated to obtain and plot various QoS network performance metrics. This framework generates and runs simulations using common simulation parameters, such as velocity, mobility models, and number of nodes automatically. It also contains permits to parse through the generated flow that contain several essential metrics a researcher may need in order to analyze the simulation, such as end-to-end delay, lost packet, and throughput.

Keywords: Simulation Framework Design, Manet, Network Simulator, NS3, Flow Monitor.

I. INTRODUCTION

Since computing devices networks are becoming more expansive and complicated, the require for reliable and scalable simulator technology is crucial. Regardless of evolution to large scale simulations for wireless network, simulations play an essential part when it comes to scalability, reproducibility, and quick prototyping in academic wireless network research [1].

The simulation-based method could be studied in detail with changing data applications and field conditions. It will also provide useful and exploitable output data. However, this method does not scale well when the complexity and scope of the networks increases. The accelerated evolution of communication networks output that has been observed over the past decade makes this method time consuming, if not impossible for large-scale implementations.

Consequently, effective simulation approaches for such networking models have turn out to be an important problem. You can find many network simulators (open source, commercial), that offer different features [2], such as NS3 [3], NS2 [4], NetSim, OMNeT++, OPNET, QualNet, J-Sim [5], and REAL.

NS3 is amongst the most extensively utilized network simulators [6], and it is commonly utilized for academic research due to its extensibility (since it is open source) and free online documentation. NS3 is quite popular for routing simulations and ad hoc networking research. The platform provides support for several protocols such as UDP, TCP, and numerous different routing protocols. We can also define network topology and the scenario that is executed by the NS3 modules. NS3 even enables users to monitor various network events as well as all network traffic, which is recorded using the tracing module during the simulation.

Tracing technique is one of the most essential mechanisms in NS3, as it assists the researcher in gathering statistics that capture the behavior of the simulation and producing an output for further study. It supports different file types, such as Trace files or Flow Monitor files. This study will use Flow Monitor files, since Trace files typically require a considerable amount of disk space and the simulation performance is degraded by I/O operations. On the other hand, however, Flow Monitor files calculate statistics during simulation runs. One experiment [7] concluded that using Trace files can make simulations take up to 6 or 7 times longer when compared to the Flow Monitor approach, where statistics gathering occurs while the simulation is in operation.

The flow Monitor file generated using the flow monitoring module makes collecting and saving the network performance data much easier. The Flow Monitor automatically detects all flows passing through the network and then stores it in an XML file. The Flow Monitor relies heavily on the NS3 Callback and Attribute mechanisms for the simulations wireless network. Nevertheless, the analysis of Flow Monitor files can be very frustrating when attempting to get results from a scaling simulation.

Manuscript received on February 27, 2020.
Revised Manuscript received on March 14, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Halim Berradi*, SSL Lab, ENSIAS, University of Mohammed V, Rabat, Morocco. Email: halim.berradi@um5s.net.ma
Ahmed Habbani, SSL Lab, ENSIAS, University of Mohammed V, Rabat, Morocco.
Mohammed Souidi, SSL Lab, ENSIAS, University of Mohammed V, Rabat, Morocco.
Hicham Amraoui, SSL Lab, ENSIAS, University of Mohammed V, Rabat, Morocco.
Nada Mouchfiq, SSL Lab, ENSIAS, University of Mohammed V, Rabat, Morocco.

The objective of this study derives from the fact that conducting a complex series of simulations smoothly and efficiently will lead to accelerated network research and enhance the overall deliverability of the results. Simulator Framework, the proposed tool for NS3 (SFNS3), accepts any of the produced

Flow Monitor files and stores them in the database for further analysis.

Afterwards, the user can retrieve a significant number of statistics concerning network performance, such as Lost Packet, End-to-End Delay, Throughput, and Jitter.

SFNS3 can be used to process all of the generated files at once. The simulation event information that occurred during the simulation is stored in the produced Flow Monitor files. Nevertheless, for the researchers of this study, the analysis of Flow Monitor files became tedious when attempting to obtain the desired results.

Thus, by developing SFNS3, the researchers wanted to offer a tool which will shorten and speed-up the simulation evaluation results. Moreover, SFNS3 designs to automate the entire simulation process by allowing the user to insert parameters describing the simulation. Throughout the development of SFNS3, we gave to users the occasion to get essential data and QoS metrics. However, a user can be concerned in exact metric that SFNS3 does not offer for the time being. Consequently, to address this, we provided the user an opportunity to execute sql request directly to the database and consult each generated file.

The rest of this article is organized as follows: Section II provide similar works that have been developed. Section III introduces SFNS3, presents its concepts, and explains the architecture challenges that occurred throughout advancement. Section IV evaluates essential elements in SFNS3s setup plus shows the main process for Flow Monitor file. Section V presents what SFNS3 provides to accomplish the evaluation of Flow Monitor files and describes all of the different functionalities. Section VI illustrates actual uses and results. The study will then be concluded in Section VII by providing perspective to the results.

II. RELATED WORK

Equivalent software have already been created that generate statistics related to network simulation performance. Such as NS2 [8] for MANET, and [9] for WiMAX networks. Trace Graph [10], [11] is tool evaluation of trace file based on NS2 architecture, offering several analysis choices, with many different charts and statistical reports. It is developed in MATLAB [12] and could be assembled to execute without MATLAB. it provides also to the capacity, to get valuable statistics within a graphical user interface from a given NS2 trace file. The set of statistics contained QoS network metrics. In addition, it generates both 2D and 3D graphs for metrics for example: mean sums, delay, throughput, jitter.

JTrana citeqian2008jtrana is NS2 wireless trace analyzer developed on Java used to study NS2 wireless simulation using a graphical user interface. JTranas functions include the production of overall network information as well as the plotting many charts performance metrics. It can handle wired and wireless trace files format. besides, you can save the output trace file to MySQL database for exploitation later.

Similar to JTrana, JDNA [13] is developed to analyze and process the trace file using the NS-2 simulator. It can parse and plot the results output, the biggest drawback for in this tool it comes with four performance metrics, and this is a limited performance metrics.

NS2 Trace Analyzer [14], [15], [16] is developed in C++, and it is compatible with all OS. Similar to the prior tools, NS2 Trace Analyzer could be utilized to receive popular network data based on the trace file output in the simulation process. However, this tool does not allow users to produce graphs about the data retrieved from the simulation. These tools offer valuable data concerning a particular scenario. Consequently, all metrics and outcomes belong to one specified situation. Additionally, these tools do not offer the user to manipulate outcome files. To obtain data concerning a different simulation, the user opened the resulting output separately, and it is a slow process, time-consuming, and it could be suitable for small files. However, in terms of simulations that generate big files, this approach could be considered inefficient. Furthermore, It is up to him to keep them well prepared and secure.

When it comes to efficiency during trace file loading, the aforementioned tools act effectively for small date output. However, the performance deteriorates significantly, in terms of severe simulations that generate trace files within 100 MBs.

Lastly, J-Sim [17], [18] is a Java tool, free, and open source simulator centered around the concept of Autonomous Element Structure. Consequently, each element can be independently developed. Furthermore, the method elements communicate in terms of information exchanges is particular. J-Sim is platform independent due to its programming language. Aside from that, J-Sim also could utilize in addition to script languages such as Python or Tcl in NS2.

The authors of this study [19], shed light on how to benefit from v.2 of the network simulator so that it can reduce the complexity of its use for university students, since comparing the existing tools was not enough. The authors of this study developed a tool called GUINs2 in order to allow students to use NS-2 even if they have no experience working with scripts and line orders.

So, we founded many tools proposed in the literature, such as NS2 Trace Analyzer[16], Trace Graph[15], and JTrana [20] provided users the ability to evaluate MANET performances and generating numerous statistics, measurements, and charts. Otherwise, we found that they leak NS3 support, extensibility, multi-threading, limited metrics, limited statistics graphs. In our work, we will try to propose a solution named SFNS3 that will help us to solve all those issues.

III. SFNS3 OVERVIEW

SFNS3 is designed to provide an analogous function however it is focused on NS3 instead of NS2.

SFNS3 allows execute a script that containt the information about all the simulation context in a GUI. The simulation input utilised to create Flow Monitor records that are eventually subjected to evaluation akin to the processes of the previously mentioned simulation evaluation tools. The overall objective of this study is to propose a tool run an entire simulation and its evaluation in a very straightforward, simple, and strong manner.

SFNS3 is a tool originally created to help analyze simulation of network, flowMon files as well as their structural design in Fig.1, SFNS3 is a multi-tenancy, three-tiered architecture: user interface presentation layer, calculation and processing company layer, and data storage database layer. In SFNS3, these are distinct layers. The business layer itself is indicated as the core. In addition, SFNS3 follows the architectural pattern of Model-View-Controller (MVC). For performance considerations, the database server and the SFNS3 server can be placed on the same machine or distributed across separate computer servers.

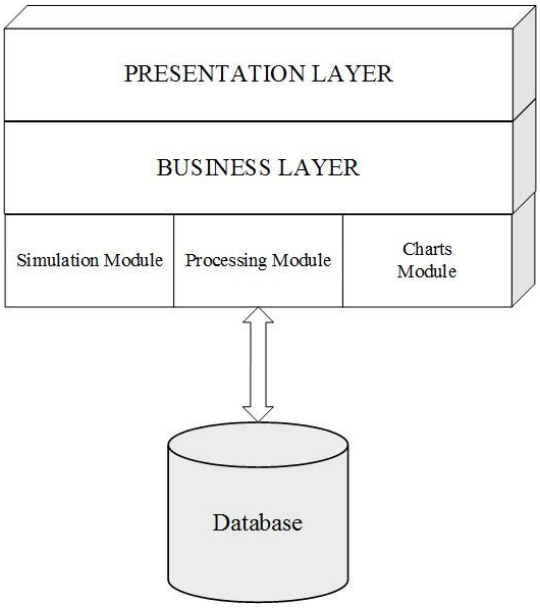

Fig. 1 SFNS3 architecture.

In this section, along with how they eased, an assessment of the design problems that emerged during SFNS3 development will be provided. In addition, the architecture of the instruments will be evaluated as it was created on the basis of the goals it was intended.

A. Issues with the design

One of SFNS3's primary objectives is to help analyze the performance of network protocols, using tracing files or flowMon files is one of the techniques used to achieve metric performance. We concentrated on NS3 flowMon documents produced through the Flow Monitor module in this research. The objective of the module is to provide a flexible scheme for measuring network protocol efficiency. The module utilizes probes mounted on network nodes to monitor the node exchanged packets and will evaluate multiple metrics.

Packets are split by their respective flow. Where every flow define using the flow tuple: protocol, source (IP, port), destination (IP, port). We can save the data collected for each flow in XML format with the extension flowMon. In addition, to request particular statistics about each stream, the user can access information directly. The primary output of the model is formatted XML report concerning flow statistics. An illustration of the same is done in Fig. 2.

```
<?xml version="1.0" ?>
<FlowMonitor>
  <FlowStats>
    <Flow flowId="1" timeFirstTxPacket="+2386.0ns"
        timeFirstRxPacket="+2390064.0ns"
        timeLastTxPacket="+99886.0ns"
        timeLastRxPacket="+99887131.0ns"
        delaySum="+118191.0ns" jitterSum="+5028130.0ns"
        lastDelay="+286105.0ns" txBytes="35972"
        rxBytes="35972" txPackets="391" rxPackets="391"
        lostPackets="0" timesForwarded="0">
    </Flow>
    ....
  </FlowStats>
  <Ipv4FlowClassifier>
    <Flow flowId="4" sourceAddress="10.1.1.5"
        destinationAddress="10.1.1.1" protocol="17"
        sourcePort="49153" destinationPort="9" />
        ....
  </Ipv4FlowClassifier>
  <Ipv6FlowClassifier>
  </Ipv6FlowClassifier>
</FlowMonitor>
```

Fig. 2 FlowMon XML structure.

Starting with FlowMonitor::FlowStats, the flowMon File has XML architecture for each flow with a unique identifier that defines the flow that has occurred, preceded by data about that flow. A further comprehensive description of the individual characteristics in the data structure flow follows. The following characteristics can be found in FlowMonitor::FlowStats for each flow: timeFirstTxPacket, timeFirstRxPacket, timeLastTxPacket, timeLastRxPacket, delaySum, jitterSum, lastDelay, txBytes, rxBytes, txPackets, rxPackets, timeForwarded, lostPackets. There was the same structure in every flowMon file. Consequently, coding the Flow file structure within the source code instruments could be a very simple way to implement the flowMon file recognition process. This way, upon a flowMon file being given as an input, it can be possible to match its configuration and save it. In addition, the previously mentioned flowMon file format contains some extra tags depending on the configuration of the simulation situation and the types of packets that pass through the simulated network. Finally, another problem that emerged during SFNS3 design and execution was the transfer of the outputs data for storage in the database. It is usual for a simulation to produce a large size output files, and as a result, it becomes a critical problem to handle all the generated files. SFNS3's implement flowMon file detection and processing after that, it save the output data to the database using python that is highly effective in those scenario. we try to provide the lowest possible transfer time to the local database.

IV. SFNS3 ARCHITECTURE

As can be seen in Figure, SFNS3 is based on three-layer architecture. 1 The primary level is that of User Interaction Management Presentation Layer.

The layer makes use of the graphical interface whereby the user can communicate with SFNS3, the layer of presentation transfers the requests of the user to the layer of business as a platform for real processing and calculation.

The Business Layer takes charge of the demands of the user and performs the entire procedure. The Business Layer comprises of several distinct modules, each of which is accountable for a distinct job, to enable all the distinct activities that SFNS3 conducts. The Main Processing Module is accountable for identifying, parsing, and storing file flowMon, which is one of the key processes for SFNS3. It is also responsible for the calculation of the General Simulation data and the General Node Information, and it is the responsibility of the Charts module to plot each type of chart. It is the responsibility of the Database access layer to store data in the local database or to return the information demanded by the Business Layer. The access layer of the Database only interacts with the form of the Business layer which either gets data for storage database.

A. Database

SFNS3's Database use a PostgreSQL as Database Management System (DBMS) where all flowMon file data is stored. The database includes several tables to save the processed the flowMon documents and all the gathered data from the simulation. Apart from the tables with flowMon file data, another table exists that contains the performance metrics shaped out when a flowMon file is loaded into SFNS3. Thus, every time a user loads the results of simulation from the database, it is not necessary for the tool to create the measurements. Conversely, as with the data from the flowmon file, all measurements offered by Flow Monitor are loaded from the database so that there is a reduction of the loading time. One of SFNS3's primary advantages is multithreading, which enables various simulations to be run simultaneously. All simulations are running in the system as a thread. Multi-threading, therefore, leads through multitasking to the greatest usage of the CPU. In a multiprocessor architecture, using multithreading, each thread can operate on a distinct processor in parallel, and this increases system competitiveness in direct comparison to a single processor scheme, where only one thread can operate on a processor at a moment, and it is more cost-effective to use threads as they distribute processor resources more efficiently.

The author discusses the most remarkable and unique characteristics of SFNS3 in this chapter and discusses this tool's structure and core.

In various applications, the XML file was used to organize the content generated in a structured format. It is observable that the exploitation of the XML file is very common and results in more vigorous and adaptable apps. On the other hand, SFNS3 uses flowMon XML file in the various steps of generating, processing and analyzing stored data.

Each flowMon file has to be properly read. So, All generated flowMon files will be processed by SFNS3. If we have an empty SFNS3 file, we skip it. Fig 2 describes the structure of a flowMon XML, and displays a number of flow lines in a XML format. The distinct fields for each Flows are based on the following fields: timeFirstTxPacket, timeFirstRxPacket, timeLastTxPacket, timeLastRxPacket, timeLastRxPacket, delaySum, jitterSum, lastDelay, txBytes, rxBytes, txPackets, timeForwarded. These lines contain information about the actual flows of a flowMon file as described by the flowMon manual [7]. We use the class showed in Fig. 3 to save the information that we gather from the flowMon File.

```
-name flowId -type number
-name timeFirstTxPacket -type number -name timeFirstRxPacket
  -type number
-name timeLastTxPacket -type number -name timeLastRxPacket
  -type number
-name delaySum -type number
-name jitterSum -type number -name lastDelay -type number
-name txBytes -type number -name rxBytes -type number
-name txPackets -type number -name rxPackets -type number
-name lostPackets -type number
-name timesForwarded -type number
```

Fig. 3 Flow class.

B. FlowMon file recognition process

The processing of flowMon files is one of SFNS3's most critical processes, and the manner it has been applied distinguishes SFNS3 from other comparable proposed solutions. SFNS3 allows users to extract, readings, calculate and plot graphs from flowMon files. To analyze the information in a flowMon file, you must first read the XML data file. The general idea of the system shown in Fig. 4 for analyzing, identifying and storing a flowMon information. And there's a Fig. 5. The rest of this subsection is defined The method of identification of flowMon files is the operation in which SFNS3 tries to read the specified flowMon file as an input.

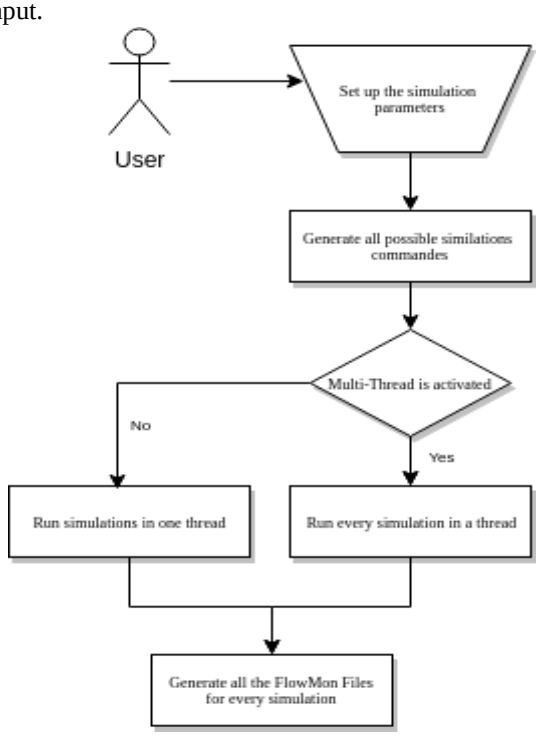

Fig. 4 Simulation running steps.

SFNS3 loaded All flowMon files one by one, after that, it get the characteristics in each flow with the corresponding fields.

This process remains until the end of all flows of the flowMon file. If there is no flow, the flowMon file will be ignored.

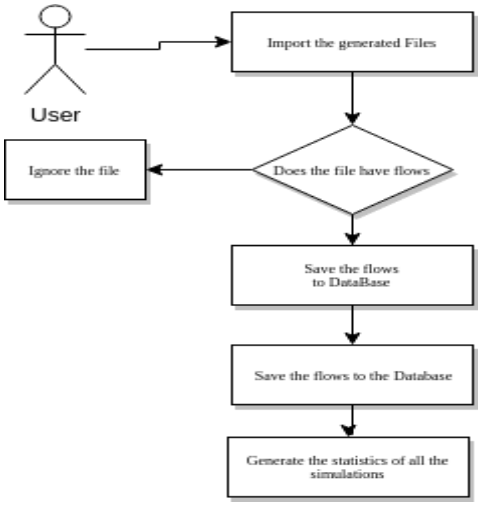

Fig. 5 FlowMon file identification and data transfer procedure.

All the data provided in the simulation input such as the number of nodes, sink number, used protocols, used mobility models, speeds, pause times, and other fields. All the fields related to simulation data presented in the class shown in Fig. 6. All flows are loaded and the data process begin to a designed table in the database. This table have as many columns as all the different fields identified in the flowMon file. If the flow lines have more fields than the standard structure described by flowMon, then the data process ignore those fields without break the execution.

For all remaining files, SFNS3 repeat this job until all produced files are treated. The data transfer mechanism following the reading of the flowMon file as described above. It is one of SFNS3's most significant modules and therefore the author in Algorithm 1 provides a more comprehensive explanation of it.

V. SFNS3 ANALYSIS

This section offers a overview of SFNS3's key processes and features. An overview of earlier analyzed problems (i.e. classification of a flowMon file) and the new features that offered only by SFNS3 to automate the full simulation process. SFNS3 is not only a tool for processing and storing flowMon files, it also paves the way for a user to launch NS3 scripts. To run the simulation so far, the researcher had to run NS3 and then use script bash or spreadsheet software to manually obtain and draw data. SFNS3 enables the user to select the NS3 script that he wishes to perform, provide it as an input, and then SFNS3 invokes it. An analysis of the resulting file is done by bringing into play the entire procedure explained in Section IV. Through that, the way the resulting files are generated, parsed, an analysis is done and the files are stored in the database without any further concern of the user.

A. Simulation execution

The first step is to begin parsing the output files from the execution stage of the simulation and load the produced file that matched the overall configuration of the simulation. After this task, the flowMon file will be read as an XML file, parsed based on the flows, and when we eventually get to the end of the file, the processing of information will commence. Finally, the data about the produced documents saved in a particular table must be placed in the database upon the completion of the processing and parsing stage.

```
-name ns3DirectoryPath -type char
-name outputPath -type char
-name simulationDirectory -type char
-name bonmotionFileTemplate -type char
-name protocol -type list
-name speed -type list
-name pauseTime -type list
-name mobilityModel -type list
-name nodeNumber -type list
-name sink -type number
-name repetitionNumber -type number
-name threadNumber -type number
-name totalTime -type number
-name simulationFiles -type one2many ref "flowMon file"
```

Fig. 6 Simulation information class.

B. Statistics calculation

Procedures of processing and storage for flowMon records proceeded by the stage in which SFNS3 computes all the simulation documents collected. These metrics are automatically computed without involving any user.

1) General simulation information: The overall simulation data calculated for the input includes data such as simulation duration, the number of threads, the number of nodes, the number of sinks, the protocols chosen, the used -name simulation -type many2one "Simulation Information class" -name flowNumber -type number name timeFirstTxPacket -type number -name timeFirstRxPacket -type number

```
-name timeLastTxPacket -type number -name
timeLastRxPacket -type number
-name delaySum -type number
-name jitterSum -type number -name lastDelay -type
number
-name txBytes -type number -name rxBytes -type number
-name txPackets -type number -name rxPackets -type
number
-name lostPackets -type number
-name timesForwarded -type number
-name meanDelay -type number
-name meanJitter -type number
-name meanTransmittedPacketSize -type number
-name meanTeceivedPacketSize -type number
-name meanTransmittedBitrate -type number
-name meanReceivedBitrate -type number
-name meanHopCount -type number
-name packetLossRatio -type number
-name packetDeliveryRatio -type number
-name throughput -type number
```

Fig. 7 Simulation output class.

models of mobility, the speeds values, the pause times. The used NS3 simulation file in the scratch directory is adapted to receive all those values as parameter.

Lastly, the following additional information is presented for flowMon files: the number of flows generated and bytes transmitted as well as the number of bytes received for each flow, the lost packet and all the information supported by flowMon file as shown in Fig. 8. This allows the user to get rapidly other data about the ran simulation.

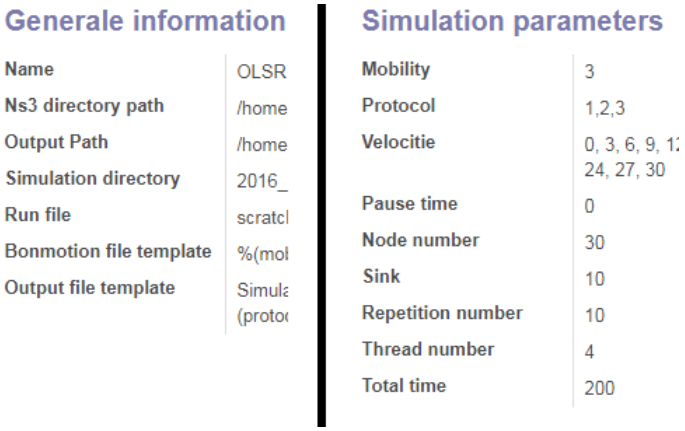

Fig. 8 Simulation main view.

2) Node specific information: Apart from the general simulation data that is calculated by SFNS3, the user will be able to get other specific information on a particular simulation node. After the flowMon file Processing is carried out, and the simulation data user can derive matching information through the collection of a a specific node. Such information includes statistics such as Number of Sent, Received, Packets and bytes forwarded and the number of generated packets for that node.

Algorithm 1 Simulation execution algorithm.

```
initialisation:
mobilities ←mobilitiesList
models ←modelsList
protocols ←protocolsList
nodeNumbers ←nodeNumbersList
velocities ←velocitiesList
pausetimes ←pausetimesList
Begin
for mobility in mobilities do
   for model in models do
      for protocol in protocols do
         for velocitie in velocities do
            for pauseTime in pauseTimes do
               for nodNumber in nodeNumbers do
                  runSimulationOf(mobility,
                  model, protocol, velocitie,
                  pauseTime, nodeNumber)
               end for
            end for
         end for
      end for
   end for
end for
```

3) QoS parameters: QoS parameters can also be extracted via SFNS3 are delivery rates in packets and throughput, minimum and maximum of end to end delay, average end to end delay, average jitter delay, minimum and maximum of jitter delay, and ratio of lost packet. The results showed after all the suitable calculation have been execute. If no communication exists between the nodes, no flowMon file can be generated by the simulation.

4) Plotting charts: The researcher can plot various graphs based on the saved data after the process stage of flowMon file. A graph plot activated by default via the "Results" button in the SFNS3 simulation input primary interface as showed by Fig. 9.

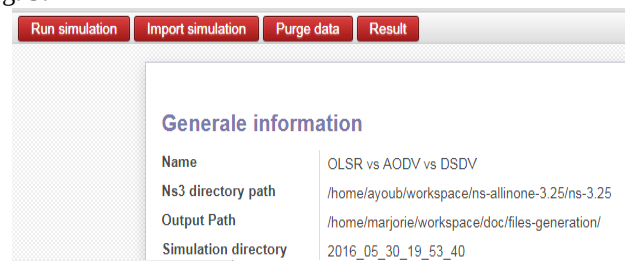

Fig. 9 SFNS3 actions.

The primary benefit of SFNS3 is that the graphs provided are unlimited; you can see the mix of chosen protocols, number of nodes, speeds, mobility models and QoS metrics such as, delivery rate as well throughput, jitter delay, and packet loss ratio. Drawing of the charts can be done using linear charts or bar charts or the simulation results can be exported by the user as CSV file, taking advantage of other drawing tools. Metrics presented in Fig. 7 can be chosen as the QoS to make a drawing of a chart as presented in the Fig. 10. After setting, then the plotting of the chart is done automatically.

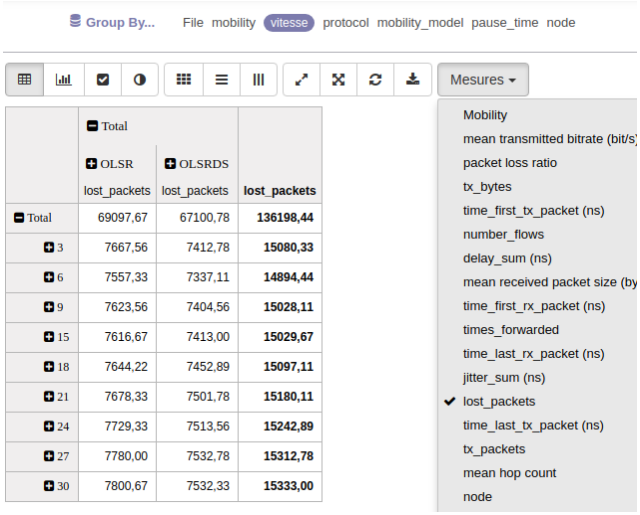

Fig. 10 Change the metric to lost packet.

C. User initiated SQL queries

SFNS3 has a database table in which the information of all the flowMon file is stored, the table structure is based on the flowMon XML structure. The table columns are as many as the XML file's fields. Having continuously stored all the data of each produced file in the database paves the way for their reuse on demand without having to reload again.

Although the general simulation data and metrics generated by SFNS3 cover up the most prevalent statistics that a user may want about simulation, there is always the case that someone may show interest in a other specific metric that SFNS3 is not currently offering this is why we add to the user the SQL inquiries directly to the database.

The user can defined the simulation parameters and indicate the script for simulation to be tested, and SFNS3 would have execute all the stages shown in Fig. 4 And Fig. 5. So, SFNS3 can run the entire operation and return all the results in a way that is user-friendly, robust and compact. SFNS3 thus seeks to promote the need for a more robust and programmed strategy to facilite the simulation process.

VI. SFNS3 USAGES AND RESULTS

In this section, we present SFNS3 utilization sample by opening a new simulation and showing the data, the tools can obtain. The authors present the computation of simulation data input, and the generated graphs that can be obtained using SFNS3. Table- I offers the parameters of simulation that we used in our experiment, there were 30 wireless nodes in the situation used to build the sample simulation.

Table- I: Simulation setups.

Attributes	Values
Physical and MAC layer	802.11b
Number of nodes	30
Mobility Model	ManhattanGrid
Simulation Time	200s
Packet size	256 bytes
Protocols	OLSR, OLSRDS
Speed	3,6,9,15,18,21,24,27,30
Transmission Range	40 m
Area	1000m X 1000m
Simulator	NS3

The generated flowMon files generated by SFNS3 will be processed as outlined in Section IV were used to make all the hard work for us. When analyzing and processing the flowMon file, its contents are viewable from a interface view as shown in Fig. 8, another characteristic distinctive to SFNS3 is to allow users to get all the information they might require made available and ready to use. The views are developed based on the XML file used to define the flowMon file and process it. In addition, it also provided the overall simulation data composed of statistics on the scenario.

As well as additional fields concerning the simulation information such as the number of flows, the overall data can be regarded. The readings in the tools are automatically after the successful transfer to the database of the flowMon file.

A. Simulation parameters extraction

Once a flowMon file is added to SFNS3 or encumbered from the catalog, multiple QoS parameters like Throughput, End to End Delay, Jitter, and Packet Loss Ratio can be obtained. These parameters are the most useful to study network simulation efficiency.

The rate of Packet delivery is calculated through the dividing of the number of packets sent and received effectively between the nodes chosen at the moment the first and last packets arrived between the nodes. In order to calculate the throughput, we split the number of bits effectively sent and received between the chosen nodes at the moment of entry of the first and last bit. An evaluation of the End to End Delay for each packet between the two nodes is carried out through the calculation of the time interval between the moment of transmission and the moment of entry of each packet passed from the source node to the destination node. The Jitter associated metrics are calculated on the basis of the RFC 3550[21]. The ratio of Packet Loss is evaluated through the calculation of all the packets sent and received between the two nodes, dividing their difference by the packets sent and multiplying by 100.

B. Chart plotting

A useful functionality provided by SFNS3 is to draw different graphs based on the processed data, as shown in Fig. 11, Fig. 12 Of course, and Fig. Fig. 13. The graphs can refer to either to the evolution of QoS metrics (such as Packet Delivery Rate, Throughput, Delay Jitter, and Packet Loss Ratio) in relation of node number, velocities, mobility model. There are innumerable combination of graphs that can be obtained using SFNS3.

	Total		
	OLSR	OLSRDS	
	lost_packets	lost_packets	lost_packets
Total	69097,67	67100,78	**136198,44**
3	7667,56	7412,78	**15080,33**
6	7557,33	7337,11	**14894,44**
9	7623,56	7404,56	**15028,11**
15	7616,67	7413,00	**15029,67**
18	7644,22	7452,89	**15097,11**
21	7678,33	7501,78	**15180,11**
24	7729,33	7513,56	**15242,89**
27	7780,00	7532,78	**15312,78**
30	7800,67	7532,33	**15333,00**

Fig. 11. Lost packets over speed in pivot type.

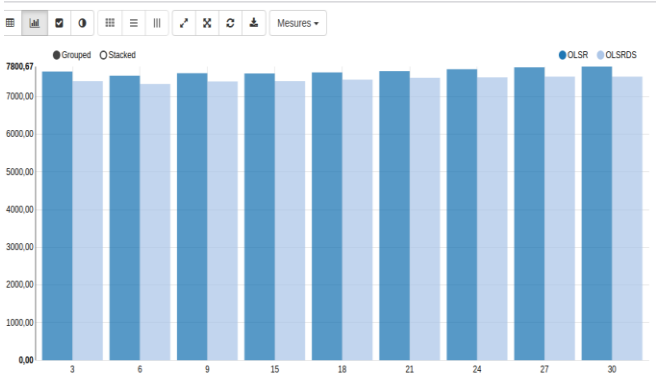

Fig. 12. Lost packets over speed in bar chart type.

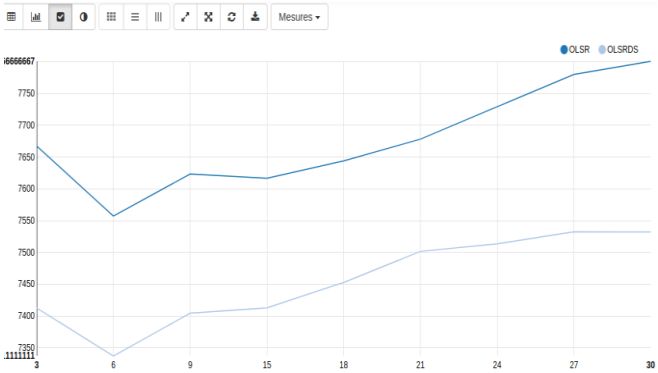

Fig. 13. Lost packets ratio over speed in line chart type.

In addition, we can permit between x-axis and y-axis in the plot area. Finally, the user can change many chart parameters and see the immediate results for each choice of parameters. Figure shows an instance of selecting a metric to recover all lost packets from the Fig. 10.

C. Executing NS3 scripts via SFNS3

Every scenario of NS3 simulation is defined and built using the language of C++ scripting. A user can build simulation scenario that test protocols performances in application context. The user must invoke the script as a parameter after generating the C++ script for it to be executed. Upon completion of the simulation, the user must run the flowMon files process intervention to automatically get the final outcomes. While these are the steps to perform a specific simulation situation, SFNS3 allows the user to execute a C++ script and carry out all the simulation scenario combination, after that, locate the resultant flowMon file, and begin the flowMon file analysis process.

D. SFNS3 results

In this section, we make a presentation of timing outcomes on the processes outlined in Section IV, including recognizing the flowMon documents, parsing, processing, and ultimately shifting their data to the database. We have shown that SFNS3 does not only manage to handle all distinct types of flowMon files by using flowMon, but it does so extremely fast. Therefore, we present outcomes for a multitude of flowMon file dimensions in Figure 14 concerning the above processes.

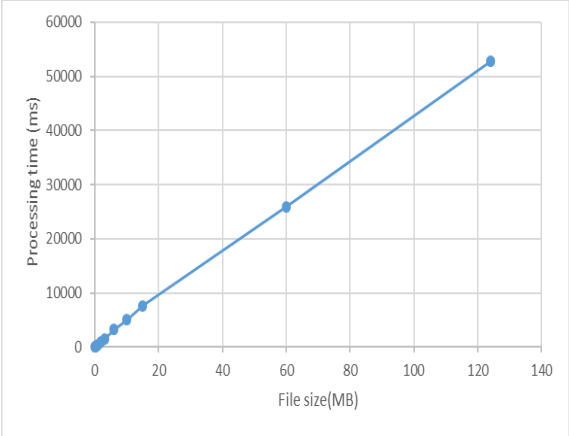

Fig. 14. Processing time results to FlowMon file.

Based on previous outcomes, it is clear that SFNS3 manages to maintain their time of processing exceptionally low even for large flowMon files. For flowMon files up to 10 MB in size, the time of execution is minimal and does not increase significantly, although the processing time starts to grow after an almost linear increase for flowMon file sizes exceeding 10 MB. We also present a comparison between SFNS3 and other common simulation analysis tools: Tracegraph and jTrana tools have been described in Section II, they are among the most commonly used to process simulation output fils, but one of their primary drawbacks is the amount of moment they need to open the file [22]. As we have shown above, in this assignment, SFNS3 performs extremely well. A comprehensive comparison between these tools and SFNS3 is therefore presented in Figure 15:

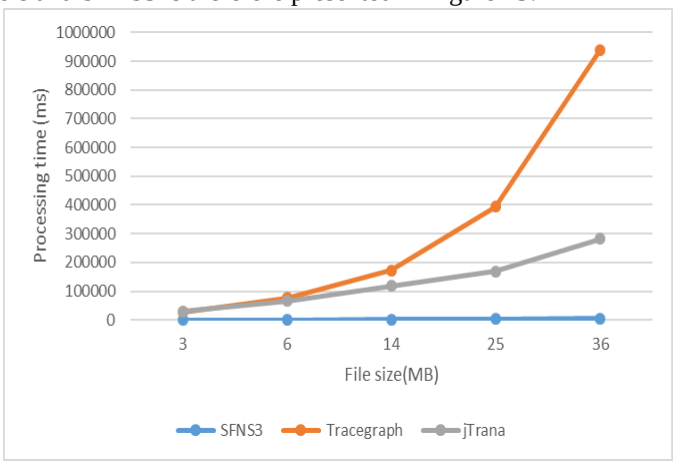

Fig. 15. File processing time comparison.

The size files produced in the wireless simulations for all three tools were the same. We physically calculated the time it took for each tool to open the file. In addition, we repeat the measurements multiple times and calculated the mean value to achieve more precise outcomes. As it can rapidly deduce from the above figure, SFNS3 produces timings that are significantly lower than the other two tools. This fact gets to be more evident if we reflect on that the most important processing time for SFNS3s is for the 36 MB file and even that the time for processing time is smaller than the time it takes for the other two tools to process the 3MB size file.

VII. CONCLUSION

In this paper, we proposed a simulation tool called SFNS3 which seeks to automate the process simulation from the execution of NS3 scripts until analyzing the results and generate the graphs. To achieve these goals, SFNS3 introduces the new idea that allows the users to rendered and manipulated their simulation. In addition, one of SFNS3's enhancement is to speed up the entire simulation process from configuration simulation, process stage, and generate distinct simulation graphs. During this task, however, similar tools were not very productive and required a fair amount of time to collect the result from all the files generated.

As we described carefully in Section VI, the use of flowMon allowed SFNS3 to perform this job considerably (up to 100 times quicker) decreases the time process compared to other popular tools.

Retrieval Number: F9146038620/2020©BEIESP
DOI: 10.35940/ijrte.F9146.038620
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Another specific feature of SFNS3 is the capability to amass all simulation files that have been generated in the database instead of handling theme directly from the disk. In addition, SFNS3 helped to extract a range of simulation QoS statistics, and graphs and export all the simulation results in CSV or Excel format. In plus SFNS3 enabled the execution of C++ simulation scripts and keep the generated files organized in folder without further user involvement.

For now, SFNS3 is able to offer an easy-to-use framework for simulations to graph analysis, but the C++ scripts must be build using NS3 before launch the simulations process. However, many researchers who are not acquainted with C++ and NS3 may want to use a straightforward interface to prepare easy typologies for testing distinct situations without learning all the distinct parameters needed by NS3 parameters. In that sense, we will work towards the improvement of the user interface from which to design a simulation interface. Using the techniques already provided by SFNS3, a C++ script can be prepared and performed. Since SFNS3 is elaborated in a modular approach following the main beliefs of object oriented programming (OOP) and using XML, we think that extending the functionality of the SFNS3 it will be a great enhancement in mobile communication fields.

REFERENCES

1. G. Chengetanai and G. B. O'Reilly, "Survey on simulation tools for wireless mobile ad hoc networks," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–7, IEEE, 2015.
2. S. Siraj, A. Gupta, and R. Badgujar, "Network simulation tools survey," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 1, no. 4, pp. 199–206, 2012.
3. Simulator Network 3, "Ns-3: https://www.nsnam.org." Accessed 14Oct-2019.
4. Simulator Network 2, "Ns-2: https://www.isi.edu/nsnam/ns." Accessed 14-Oct-2019.
5. I. Dorathy and M. Chandrasekaran, "Simulation tools for mobile ad hoc networks: a survey," *Journal of applied research and technology*, vol. 16, no. 5, pp. 437–445, 2018.
6. G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, pp. 15–34, Springer, 2010.
7. G. Carneiro, P. Fortuna, and M. Ricardo, "Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3)," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, p. 1, ICST (Institute for Computer Sciences, Social-Informatics and , 2009.
8. T. Issariyakul and E. Hossain, "Introduction to network simulator 2 (ns2)," in *Introduction to Network Simulator NS2*, pp. 1–18, Springer, 2009.
9. J. F. Borin and N. L. da Fonseca, "Simulator for wimax networks," *Simulation Modelling Practice and Theory*, vol. 16, no. 7, pp. 817–833, 2008.
10. S. Jansen and A. McGregor, "Simulation with real world network stacks," in *Proceedings of the 37th conference on Winter simulation*, pp. 2454–2463, Winter Simulation Conference, 2005.
11. C. Cicconetti, E. Mingozzi, and C. Vallati, "A 2 k· r factorial analysis tool for ns2measure," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, p. 4, ICST (Institute for Computer Sciences, Social-Informatics and , 2009.
12. J. Malek and K. Nowak, "Trace graph-data presentation system for network simulator ns," *Proceedings of the Information SystemsConcepts, Tools and Applications (ISAT 2003), Poland*, 2003.
13. A. M. M. Habbal, S. Hassan, and A. M. Jabbar, "Jdna: Java-based ns-2 analyzer," *Wulfenia J*, vol. 19, no. 9, 2012.
14. S. T. U. Guide, "The mathworks, inc. mail 3 apple hill drive natick, ma 01760-2098," 1995.
15. A. U. Salleh, Z. Ishak, N. M. Din, and M. Z. Jamaludin, "Trace analyzer for ns-2," in *2006 4th Student Conference on Research and Development*, pp. 29–32, IEEE, 2006.
16. P. Machan and B. Bartosz, "The trace file analysis tool trace analyzer: http://trace-analyzer.sourceforge.net." [Online; accessed 14-Oct-2019].
17. A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, "J-sim: A simulation environment for wireless sensor networks," in *38th Annual Simulation Symposium*, pp. 175–187, IEEE, 2005.
18. J. Hou, "J-sim: https://www.physiome.org/jsim/." [Online; accessed 14-Oct-2019].
19. P. Truchly and L. Humeny, "Intuitive graphical interface for network simulator 2 supporting trainings in computer networks," in *2014 IEEE 12th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pp. 483–488, Dec 2014.
20. H. Qian and W. Fang, "Jtrana: A java-based ns2 wireless trace analyzer," *Disponıvel em [http://sites. google. com/site/ns2trana/]*, 2008.
21. R. Frederick, V. Jacobson, and P. Design, "Rtp: a transport protocol for real-time applications," *IETF RFC3550*, 2003.
22. R. Ben-El-Kezadri, F. Kamoun, and G. Pujolle, "Xav: a fast and flexible tracing framework for network simulation," in *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 47–53, ACM, 2008.

AUTHORS PROFILE

Halim Berradi he received the state engineer degree from the ENSAT attached to ABDMALEK ESSADI, Tangier, in 2012. he is currently pursuing the Ph.D. degree with the Laboratory of Smart Systems (SSL), at the National School of Computer Science and Systems Analysis (ENSIAS) attached to Mohammed V University, Rabat, Morocco. Her research interests include the security of smart cities and systems based on new technologies.

Habbani Ahmed is a Professor of Higher Education at the National School of Computer Science and Systems Analysis (ENSIAS) attached Mohammed V University, Rabat, Morocco and associate researcher at the laboratory: ENSIIE France .He received his Ph.D. degree in Applied Sciences in laboratories: LEC (Laboratory of Electronics and Communications) of the EMI (Mohammedia School of Engineers) attached to the University Mohamed V Rabat, and LISIF (Laboratory of Instruments and Systems of Ile de France) of the Pierre et Marie Curie University, FRANCE. His research interests include Modeling, development and implementation of Mobile Intelligent Digital System; Modeling, optimization, development and implementation of Routing Protocol; Modeling, optimization, development and big data for routing protocol; Modeling, optimization and Antennas; Modeling, optimization, development, routing and smart grid; Modeling, optimization, development and Smart wireless sensors networks.

Mohammed Souidi received an engineering degree in Computer Sciences from the National School of Computer Science and System Analysis (ENSIAS), Rabat, Morocco in 2009 and a Master degree in Sciences and Techniques from Sidi Mohamed Ben Abdellah University, Fes, Morocco in 200. His current interest is the conception/optimization of the routing protocols in the domain of the mobile Ad hoc/Sensor networks. He is currently a PhD student at ENSIAS within the Smart Systems Laboratory (SSL).

Hicham Amraoui He received his Ph.D. degree in Applied Sciences at ENSIAS within the Smart Systems Laboratory (SSL) attached to the University Mohamed V Rabat. His current interest is the conception/optimization of the routing protocols in the domain of the mobile Ad hoc/Sensor networks using Game theory.

Mouchfiq Nada was born in Casablanca, Morocco, in 1992. She received the state engineer degree from the National School of Electricity and Mechanics (ENSEM) attached to HASSAN II University, Casablanca, in 2016. She is currently pursuing the Ph.D. degree with the Laboratory of Smart Systems (SSL), at the National School of Computer Science and Systems Analysis (ENSIAS) attached to Mohammed V University, Rabat, Morocco. Her research interests include the security of smart objects and systems based on new technologies.

Retrieval Number: F9146038620/2020©BEIESP
DOI: 10.35940/ijrte.F9146.038620
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication