Network Simulation in a TCP-Enabled Industrial Internet of Things Environment-Reproducibility Issues for Performance Evaluation

Daniel Morato, *Member, IEEE*, Carlos Pérez-Gómara, Eduardo Magaña, *Member, IEEE*, and Mikel Izal, *Member, IEEE*

***Abstract*—Network simulation is a tool used to analyze and predict the performance of Industrial Internet of Things deployments while dealing with the complexity of real testbeds. Large network deployments with complex protocols such as transmission control protocol are subject to chaos-theory behavior, i.e., small changes in the implementation of the protocol stack or simulator behavior may result in large differences in the performance results. In this article, we present the results of simulating two different scenarios using three simulators. The first scenario focuses on the Incast phenomenon in a local area network where sensor data are collected. The second scenario focuses on a congested link traversed by the collected measurements. The performance metrics obtained from the simulators are compared among them and with ground-truth obtained from real network experiments. The results demonstrate how subtle implementation differences in network simulators impact performance results, and how network engineers must consider these differences.**

***Index Terms*—Computer simulation, internet of Things, reproducibility.**

I. Introduction

THE Industrial Internet of Things (IIoT) is adding large numbers of devices to communication networks. In this era, new devices are connected and old ones are removed, equipment from different vendors is added, old applications stop being used and new ones are developed, controller servers are moved from one switch to another or even to a remote site, new links with increased capacity are added, and new routing protocols are deployed, perhaps without removing old ones, all of this at a really fast pace. Any of these changes can yield consequences in network performance that are very difficult to predict. In complex systems, such as modern networks, because of nonlinearities and synchronizations, small changes can yield large-scale consequences. This phenomenon has been referred to as the amplification principle [1].

At present, it is unfeasible to predict network behavior and performance using analytical tools. This is because a large number of elements must be considered in the model, including nondeterministic behaviors (e.g., user actions). Researchers in academia have attempted approximations based on the analysis of complex systems [2] or chaotic systems [3]; however, the results are not applicable for problem solving in real networks, and engineering approximation continues to be employed in industry [4].

An alternative solution to analytical prediction is simulation modeling. New network or system configurations are tested before actual deployment using simulation software. A network administrator can predict the effects of moving localized controller servers to a cloud service by simulating the present environment and then implementing larger delay and loss parameters and network bottlenecks in the access links to simulate the future environment. New network topologies can be tested, and the results of link utilization can be evaluated. Depending on the complexity of the scenario or the expected changes, simple or complex models can be implemented in the simulation.

Several software simulators have been adapted or designed to simulate computer networks. Commercial products such as Riverbed Modeler (OPNET) [5], OMNEST [6], QualNet [7], and NetSim [8] are the most prevalent simulators in the industry. Academic researchers typically employ simulators with an open license, e.g., OMNeT++ [9] (i.e., the free version of OMNEST), ns-2 [10], ns-3 [11], or JSim [12]. These are generic network simulators, although most of them include models for wireless interfaces that are used in the IoT environment (802.11, 6LowPAN, LoRa, LTE) [13] [14], and for industrial network scenarios [15]–[17]. Software designed for other specific environments also exists [18], such as simulators for wireless networks [19], vehicular networks [20], or sensor networks [21]. Moreover, sometimes a single simulator does not provide sufficiently accurate results, and several simulators that have been optimized

Manuscript received December 16, 2020; revised April 9, 2021; accepted May 21, 2021. Date of publication May 26, 2021; date of current version October 27, 2021. This work was supported by Spanish State Research Agency under Project PID2019-104451RB-C22/AEI/10.13039/501100011033. Paper no. TII-20-5630. *(Corresponding author: Daniel Morato.)*

Daniel Morato, Eduardo Magaña, and Mikel Izal are with the Department of Electrical, Electronic, and Communications Engineering, Public University of Navarre, 31009 Pamplona, Spain, and also with the Institute of Smart Cities, 31009 Pamplona, Spain (e-mail: daniel.morato@unavarra.es; eduardo.magana@unavarra.es; mikel.izal@unavarra.es).

Carlos Pérez-Gómara is with the Department of Automatics and Computer Sciences, Public University of Navarre, 31009 Pamplona, Spain (e-mail: carlos.perez@unavarra.es).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3084128.

Digital Object Identifier 10.1109/TII.2021.3084128

for different communication layers must cooperate to obtain realistic results [22].

However, simulation-based network evaluation is not without its drawbacks. First, complex networks cannot be fully simulated because the computational cost is prohibitive. To address this problem, simulators that are capable of parallel processing using dozens of CPU cores during the simulation have been developed [23]. Second, although open simulators typically offer application programming interfaces for the development of new models and existing models can be used if they can be applied to the target scenario, no simulation model exists for every application level software and protocol. Third, user actions, or the generation of input sensor data, critically affect network traffic generation; however, these data are typically not predictable, and either recorded traces or stochastic models must be used to reproduce their behavior.

Unfortunately, there are no generic models for user actions, and simulators typically only offer simple random value generation functions. Finally, the results obtained using a simulator are not easily reproducible by a different one. This is because the simulated network scenarios contain a large number of protocols and mechanisms that could not only be implemented differently, but could also contain default configuration parameters that are dependent on implementation decisions.

In this article, we focus on the reproducibility problem. This problem exists in all research fields that use simulations (e.g., computer science, logistics, social sciences, and computational biology), and several panels and position papers have been discussing the relevant problems and suggested solutions for more than two decades [24]–[26]. The data networks in these simulations are complex systems, as a large number of protocols and mechanisms in these protocols interact, potentially significantly affecting the performance results. The presence of queueing systems in all of the protocol layers and the unpredictability of external inputs (user actions) increase the sensitivity of the results to implementation details, configuration parameters or seemingly inconsequential decisions applied to the design.

Although the transmission control protocol (TCP) is not the traditional transport mechanism in industrial networks, which have been dominated by closed solutions, TCP/IP network stacks are common nowadays in IIoT deployments. Popular SCADA systems offer transport over TCP, such as Modbus/TCP, IEC 60 870-5-104 or Ethernet/IP CIP (Common Industrial Protocol) systems. Most of the drawbacks attributed to TCP in these environments have been proven as not applicable anymore [27] and TCP/IP implementations for 8-bit microcontrollers are being used, with as little as 40 KB of code [28]. TCP has probably the highest implementation variability for different operating systems. Therefore, we have focused on simulators for which academic researchers have developed models for applicability to multiple versions of this protocol on the market (e.g., Reno, New Reno, Vegas, Cubic, Illinois, Westwood, HS-TCP, Fast, and Compound TCP [29]).

For the past several decades, the most popular simulation tool in research studies has been ns-2. It is the reference tool for any comparison of results, but it has not been updated consistently in recent years, its last update being in 2011. Some articles already report discrepancies obtained with it because many new mechanisms implemented in modern TCP/IP stacks are not present in ns-2 [30]. This is because of the appearance of ns-3 in 2006. However, it should be noted that ns-3 is not a new generation of ns-2; it is a new network simulator independently developed and which has only inherited a similar name.

The implementation of TCP in ns-2 has been evaluated in studies such as [30] and [31]. They reported large discrepancies in the results when compared with a real testbed. The reasons presented were the effect of mechanisms not yet implemented in the simulator and stochastic behavior existent in reality and not modeled in simulation. The implementation of new congestion control algorithms in ns-3 has also been evaluated [32], but it has been compared with theoretical expected results. The results of the studies are positive (good match with theoretical results), but they are also optimistic because the theoretical results do not take into account real complex environments but simple scenarios.

We have added OMNeT++ to the list for comparison because it is also widely used in academia; it has multiple TCP simulation models, some other simulators are derived from it [14], it offers models for IIoT scenarios [15], [16] and, although there is a version for the industrial environment that is available through a license (OMNEST), there is also a free version for academic researchers.

In this article, we focus on the comparability among simulators. We try to create exactly the same IIoT scenarios in all the simulators and we check whether a researcher or network engineer would obtain the same results and conclusions or there are significant differences in the implementations such that not only the simulations do not match perfectly the real hardware world but they do not even match among themselves. The specific versions used in this study are ns-2 2.35, ns-3 3.24.1, and version 5.0 of OMNeT++.

The scenarios simulated are representative use cases for network analysis and prediction or network protocol evaluation in IIoT scenarios. They do not intend to be complete studies of these scenarios. They are used as examples of the erroneous conclusions that can be obtained from simulation in the IIoT environment—which can be different using different software—highlight the possible sources of discrepancy and alert the researchers about the situation.

The rest of this article is organized as follows. Section II describes the IIoT network scenarios used for simulator comparison. Section III presents the performance metrics measured in these scenarios and a description of each simulation; this section also includes the comparative analysis and discussion on the causes of some of the differences. Finally, Section IV concludes this article.

II. Network Scenarios

We selected two scenarios for the simulation experiments: the "Local data" and "Remote sensors" scenarios; they were designed to recreate common problems in different IIoT networking environments. They can be applied not only to the IoT and IIoT environments, but also be used as simplifications for other networking environments such as long-distance links for

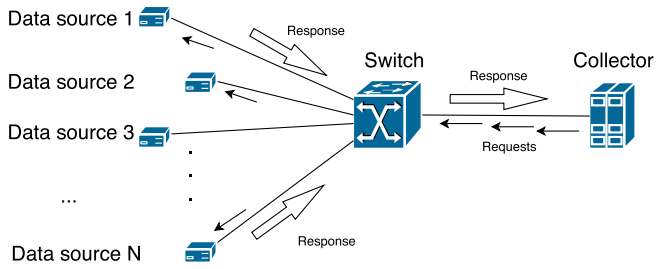

Fig. 1. TCP Incast local data scenario.

large populations of users, local area networks with transactional clients and servers or even cloud scenarios. This is not to diminish their representativeness as IIoT scenarios, but to express that similar network topologies and traffic flows can be the consequence of very different users and applications.

The traffic carried in both scenarios is TCP traffic, and we will focus on the subtle implementation details that can escalate and lead to analytical conclusions that widely differ according to the simulation software. We evaluated the different simulators using the same protocol stack configuration in all hosts. However, small differences in protocol stack or switch behavior implementation can be amplified as a result of the interactions between a large number of flows, which is common in an IIoT scenario.

We also compare the results from simulations with experimental setups, either previously published in the literature or designed for this research work.

A. Local Data Scenario

We present a simplified local area network scenario in which there is synchronization in the set of flows coming from the remote devices. These flows are multiplexed over the buffer in a switch port, typically in the direction of the port to a server collector. The result is the overflow of the switch port buffer, packets dropped, a reduction in link efficiency, and an increase in flow durations. This phenomenon has already been analyzed for IoT in a data center [33], and for data acquisition in the CERN facilities [34]. It is also a well-known phenomenon in storage and data networks with distributed applications that use the Partition+Aggregate [35] paradigm, such as the Hadoop MapReduce framework [36]. In all those scenarios, it is named as the "Incast" phenomenon. In comparison with the following remote sensor scenario, the data sources and the collector are connected to the same network switch.

We simulated the basic network scenario that results in throughput collapse by Incast. In this scenario, an application on a host (i.e., "Collector" in Fig. 1) uses a TCP connection with each one of a set of endpoints ("Data sources" in Fig. 1) to send a small distributed request, receiving a response from each data source before a new request can be issued. The data sources do not need to be the data acquisition devices; they can be the IoT gateways that communicate with the real devices using different communication technologies (for example, low power radios, or power line communications).

TABLE I
TCP SIMULATION PARAMETERS

TCP congestion control variant	Reno
Delayed ACK	Enabled (timer of 200 ms)
TCP flow control maximum window size	65535 bytes
Initial congestion window size	3 segments
Maximum Segment Size (MSS)	1460 bytes
Selective Acknowledgement (SACK)	Deactivated
Minimum retransmission timer	200 ms

TABLE II
SPECIFIC SIMULATION PARAMETERS IN THE LOCAL DATA SCENARIO

Number of data sources	1 to 25
Number of collectors	1
Transfer block size	64, 96 and 160 kB
Collector link speed	1 Gb/s
Collector link buffer size	25 to 300 packets

The time between request issuance to each source is negligible, and results in a burst of responses that overloads the buffer in the switch port connected to the collector. The losses in the switch are recovered by the retransmission mechanisms; however, because the response sizes are typically small, fast retransmission mechanisms cannot be activated, and retransmission timers typically expire. This waiting time (at least 200 ms) results in long times for response completion. The collector cannot generate the result of the distributed computation until it has received all the responses; thus, a relatively long completion time in one of them results in low utilization of link bandwidth.

The Incast phenomenon has been studied in detail and reported in literature by academic researchers [33], [34], [37], [38] and research groups within companies such as Microsoft [39] and Facebook [40]. We not only compare the results of the simulators among themselves but also with the experimental data available from [41]. The performance metric that we have selected is the number of data sources required to reach throughput collapse at the link to the collector.

Further details of the simulation parameters can be found in Tables I and II. The implementations of the TCP/IP stack in the simulators can still differ for example in the type and number of options included in the TCP or the IP header, or in the specific algorithm for adapting the delayed acknowledgement timer. There are no configurable options in most simulators to control these low-level details, therefore subtle implementation differences could result in noticeable simulation results. Many of these differences are out of the control of the researcher without a deep understanding of the inner workings of the simulator.

B. Remote Sensors Scenario

The scenario (see Fig. 2) is based on a network topology with a bottleneck link between two populations of devices (the IIoT endpoints and the data collectors). This link carries the multiplex of traffic from a large number of connections. The traffic from these connections suffers network congestion, resulting in losses at the link. The interactions among the congestion control algorithm instances were evaluated. This scenario is also very sensitive to small differences in implementation between the

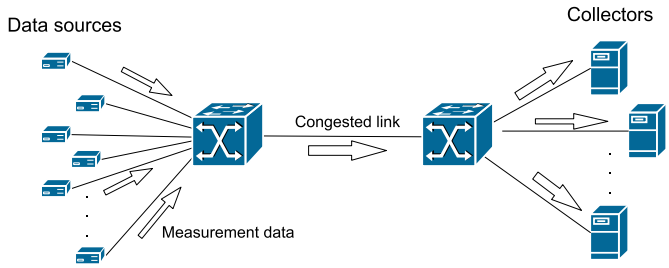

Fig. 2. Balanced dumbbell topology for the remote sensors scenario.

TABLE III
SIMULATION PARAMETERS IN THE REMOTE SENSORS SCENARIO

Congested link bandwidth	100 Mb/s or 1 Gb/s
Switching delay	20 μs
Propagation delay	5 ns
Queue discipline	DropTail
Link buffer size	From 10 to 50,000 packets
Connection arrival process	Multiplex of periodic arrival processes
Number of data sources	100
Connections per sensor per minute	From 0.6 cnx/min to 15 cnx/min
File transfer size per connection	Constant of 400 kB or Pareto ($\alpha = 1.5$, minimum of 50 kB)

simulators, and was implemented to measure the reliability of the results in performance evaluation.

We have simulated the traffic from a large population of users or applications that flow between two sites, or between a remote set of devices and a data center. As an example, in the power grid the devices are phasor measurement units and the information is synchrophasor data, collected at a phasor data concentrator or at a control data center [42]. Traffic on the long-distance link will sometimes cause congestion, resulting in losses for some connections. In an industrial local area network, Fig. 2 represents a network topology where the sensor devices are not connected to the same local area network switch as the controller hosts, therefore the traffic must be transported through an interconnection trunk link. In the power grid case, the congested link exhibits a wide area network delay, while in the industrial local area network all the links present delays in the campus network range. The data sources may be IP-based sensors or IoT gateways that aggregate measurements from devices that use non-IP communication technologies.

We sought to comprehensively compare the results of the three simulators under the conditions of the same settings and input data (i.e., connection arrival times and connection transfer sizes). The connection arrivals from each data source followed an independent periodic arrival process, all of them with the same period. They model the periodic collection of measurements from data acquisition equipment. The same random-generation seed was used in all the simulations; thus, every simulator had identical connection arrivals. TCP configuration parameters are shown in Table I and specific scenario parameters in Table III.

We created an experimental setup to compare the simulation results with ground truth. We used two Cisco Catalyst C1000-8T-2G-L switches. A pair of computers acted, one as the collector, and the other as the data sources. Both computers were connected

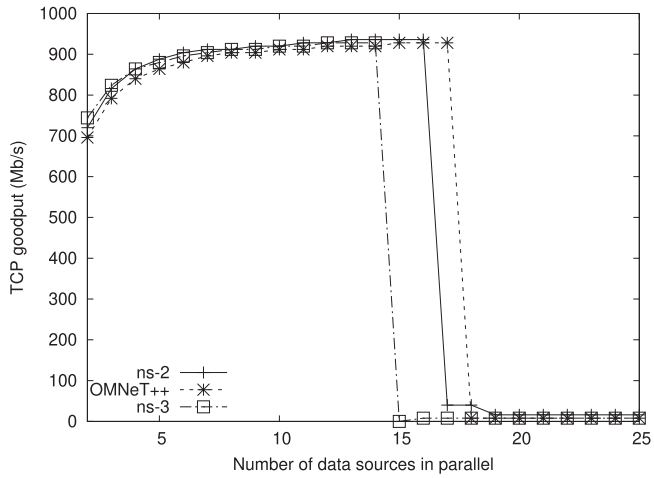

Fig. 3. Simulation results for the TCP Incast local data scenario (block transfer size: 64 kB.; buffer size: 200 packets).

through one-meter-long copper links to Gigabit Ethernet ports at each of the switches, while the interswitch link was configured at 100 Mb/s transmission speed with a 1 m copper link. GNU/Linux was the operating system installed in both computers, and the parameters in Table I were configured for the communication between sources and collector.

The performance metric that we have selected is the connection duration. This type of simulation and performance metric could be used, for example, to evaluate the time between data generation at the sources and its recollection at the central servers. The result of longer connection duration is the information being available later at the collector, delaying future decisions.

III. SIMULATION RESULTS AND DISCUSSION

In this section, we present the results for the two scenarios. We analyzed the differences in the results produced by the simulators to assess their impact and identify the reasons for the deviations. We compare the result of simulations among the different simulators, where the same results should be obtained, and also to experimental setups used as ground truth.

A. Local Data Scenario

For the local data scenario, we simulated synchronized data transfers from the data sources, increasing the number of sources until the throughput collapse took place. Fig. 3 presents the results obtained from the three simulation software packages. The abscissa axis represents the number of data sources that send a 64 kB data-block, and therefore the amount of flows that must be completed before the collector can finish its task. The ordinate axis measures the goodput that can be obtained from the link to the collector. A high goodput is obtained while the number of data sources is below a certain threshold. Above this, number of sources collapse in goodput takes place. The collapse occurs when packets are lost in the switch due to buffer overflow. These losses result in low goodput while the data source suffering the packet loss waits for the retransmission timer to expire.

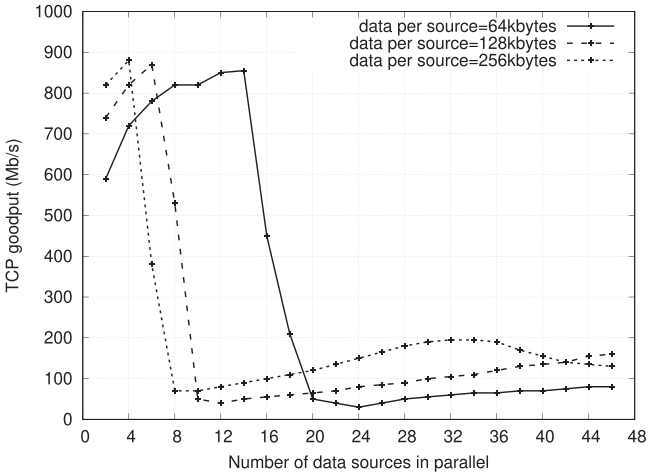

Fig. 4. Expected TCP goodput results for the TCP Incast scenario, as reported in [41]. The results are experimental, not simulated.

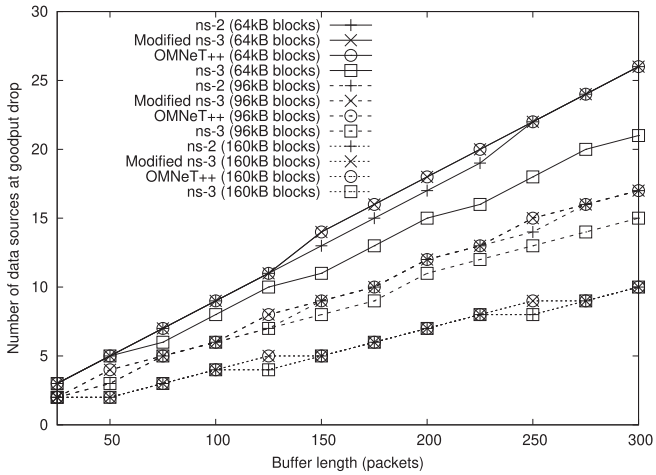

Fig. 5. Simulation results for the TCP Incast local data scenario with different switch packet buffer sizes and different block transfer size.

Ideally, the three data series in Fig. 3 should yield the same result, i.e., suffer the collapse for the same number of data sources. The configuration parameters in the scenario are as similar as the simulators allowed us to make. For example, the links were created to have the same bandwidth and delay, the same version of congestion control was applied in all of the hosts in all of the simulators (TCP Reno), the sizes of the flow control windows were set to be the same, mechanisms such as delayed acknowledgement or minimum retransmission timers that are configurable were set to have the same values, and the size of the switch buffers was the same (see Tables I and II). However, even though identical parameters and variables were applied, the final results were found to differ.

We can compare these results to a real hardware scenario using the results that Wu *et al.* [41] reported. They measured the throughput collapse from Incast in a testbed. Fig. 4 shows the results they reported after extracting the data used to create [41, Fig. 3]. In this case, "data per source" corresponds to the size of the data-block that each source sends in the response. Note that the three data plots in Fig. 3 are not comparable to those in Fig. 4. In Fig. 4, the three data series correspond to different scenarios (i.e., different block sizes). The first data series in Fig. 4 (labeled as "data per source=64 kbytes") is the one that should be the result of all the simulations.

The difference between the results of the experiments and simulations could be attributed to the buffering behavior in the switch used in the experiments [41]. In their report, they assumed that the 4 MB of total buffer in the switch were equally distributed among the 48 ports, resulting in approximately 80 kB of buffer per switch port. It should be noted that the switches used in these data center scenarios are top-of-rack switches with a large number of ports. Link speeds of at least 1 Gb/s, and small buffers implemented in the ASIC electronics [43]. However, it is known that switches can use shared buffers, and thus a port could exceed its equitable allocation. Regarding the results illustrated in Fig. 3, a buffer of 200 packets was used for all simulations, which, in the case of Ethernet MTU packets, corresponds to approximately 300 kB.

Fig. 3 shows that there are substantial differences in the implementation of each simulator that significantly affect the obtained results. For example, in this case, the conclusions differed by more than 15%, depending on the simulator. Specifically, the OMNeT++ simulator results indicated that the collapse occurs for 18 or more data sources, whereas the ns-3 simulator predicted that it would occur for 15 sources.

We studied the source code for each simulator to determine the implementation details that led to these disparities. For the case presented in Fig. 3, it was found that ns-3 deactivated the delayed acknowledgement mechanism for the first data packet received, so the rate of transfer was initially higher, with faster growth of the congestion control window; this resulted in a collapse by Incast with fewer hosts. To verify that this difference (i.e., only one more acknowledgement in the connection) leads to the differences shown in Fig. 3, we simulated the same scenario but using a version of ns-3 (i.e., "Modified ns-3") in which the code of the implementation of TCP was modified to remove this behavior in the first acknowledgement. Modified ns-3 yielded results that were identical to those obtained from OMNeT++.

Fig. 3 shows also a small difference between the series of ns-2 and the other simulators. In this case, because of the implementation of the link level and TCP options in each simulator, the feasible maximum segment size differed. One simulator uses the Ethernet header, whereas the other does not offer this option and we configured a PPP link. Furthermore, depending on the block size in the transfer, there may have been greater or lesser packets in a simulation. This led us to investigate other differences in the simulators.

We extended the simulations by varying many parameters. Fig. 5 shows the results for different transfer block and switch buffer sizes. The abscissa axis shows the size of the port switch buffer, as measured in packets, and the ordinate axis indicates at which number of hosts the collapse by Incast occurred for each scenario and simulator. For example, regarding the results presented in Fig. 3, the drop point for ns-3 occurred with 15 hosts, which corresponds to the ordinate value for an abscissa value, i.e., packet buffer size, of 200 packets in Fig. 5.

Several interesting phenomena can be observed in Fig. 5. First, the results from ns-3 do not substantially differ from those of other simulators, even with the extra acknowledgement. Second, when the size of the transfer is sufficiently large, the effects of the extra acknowledgement at the beginning of the transfer are no longer significant. These findings validate the efficacy of the simulator when the simulated scenario is implemented for a certain range of parameter values; however, these findings mask the anomalous behavior exposed previously due to extra acknowledgments, and could cause us to incorrectly assume that the results will always be valid, regardless of the scenario.

Modified ns-3 accurately replicated the results from OMNeT++, and the differences with ns-2 were found to be very small, and nearly nonexistent in some scenarios.

A conclusion for this network scenario is that all of the simulators yielded equivalent results within a reasonable error margin, but only when the small implementation difference in ns-3 had been corrected. This second aspect reveals the high sensitivity of the results to small changes in the implementation in each simulator, and confirmed that we must validate the results, or at least corroborate them with several simulators.

B. Remote Sensors Scenario

In the congested link scenario, for each TCP connection in each simulator we calculated the percentage that reflects the difference between the duration of the same connection. For example, if a connection in the reference simulator was sustained for 25 ms, whereas that in another simulator was sustained for 30 ms, then the difference was 20% when measured relative to the first simulator. We selected the connection duration because it is relevant in a measurement scenario, it is simple, easy to understand, and easy to measure in all the simulators. However, this is only one of the performance parameters that can be applied for comparison. For example, we also investigated the number of TCP segment retransmissions and reached similar conclusions.

In the top and middle plots in Fig. 6, ns-2 is the reference simulator. The results illustrate the extent of the connection duration discrepancy with respect to the reference duration (ns-2) for the same connection, using either ns-3 or OMNeT++ simulators. It can be seen that the probabilities of the durations differing from that of ns-2 results were nonnegligible.

The cumulative probability distributions of connection duration discrepancy for these simulations are shown in Fig. 7. We can see that 90% of the connection durations from ns-3 and OMNeT++ differ by more than 80% from the result produced by ns-2. Although ns-2 is typically referred to as the reference simulator in the networking literature, more modern software packages offer quite different results, which lead to very different performance predictions.

The difference between the OMNeT++ and ns-3 results were found to be minimal, with 80% of the connections having a duration that differed by less than 1%. Therefore, we can say that these two simulators yield similar results, and that their predictions are comparable, while the results produced by ns-2 significantly differ from those produced by ns-3 and, therefore, by OMNeT++.

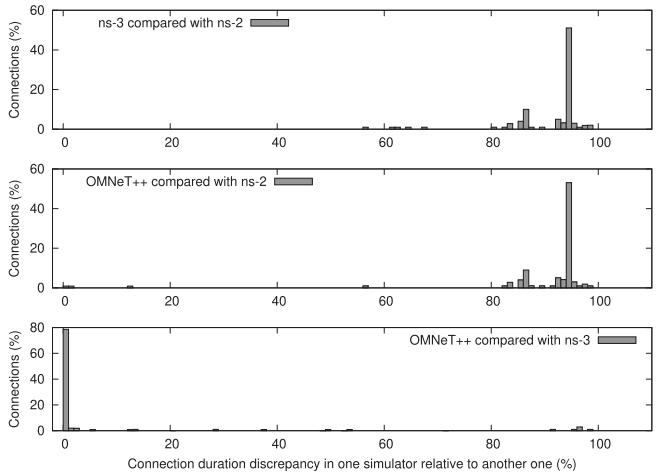

Fig. 6. Connection duration discrepancy of ns-3 (top) and OMNeT++ (middle) results relative to ns-2. Discrepancy of OMNeT++ results relative to ns-3 are at the bottom plot. Histogram of the percentage of connections as a function of percent difference. Bin size 1%. Rate of 6 cnx/min/node. Buffer size of 50 packet at the switches.

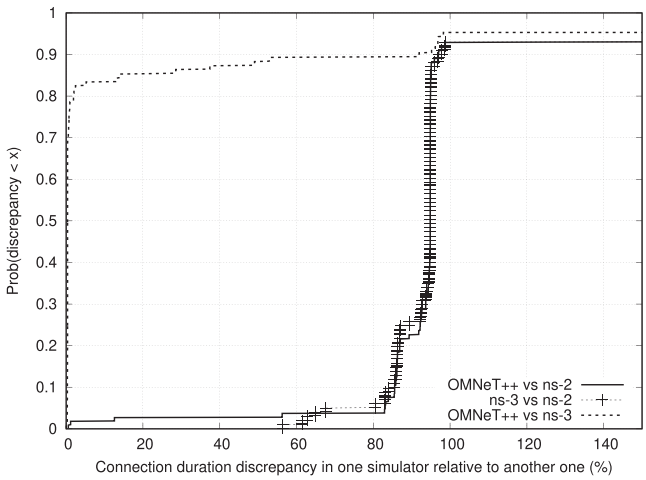

Fig. 7. Comparison of discrepancy as a cumulative probability distribution. Results of ns-3 and OMNeT++ relative to ns-2 and OMNeT++ relative to ns-3. Rate of 6 cnx/min/node. Buffer size of 50 packet at the switches.

Guo and Lee [30] offer a modified version of ns-2, including new mechanisms that obtain a better match between simulation results and the Linux TCP/IP protocol stack. We repeated the simulations using their modifications and obtained the same output. This result was expected because the Intra-Host-Back-Pressure problem that the authors solved in their modification to ns-2 is not present in this scenario, where the bottleneck is not the first-hop link from the sender (between the device and the first switch). The rest of the modifications in ns-2 are effective only when Linux CUBIC congestion control algorithm is being used, therefore they are not available when using TCP Reno, which is the version we selected for better comparability with different simulators.

We described a modification we made to ns-3 for the local data scenario. This version of ns-3 fixed the discrepancies with OMNeT++ for most evaluations in that scenario. We tested this

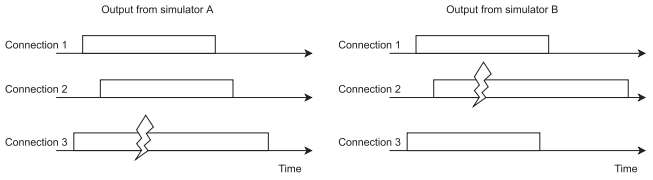

Fig. 8. Two scenarios where a different connection suffers a packet loss.

version in the remote sensors scenario but we did not obtain a significant difference in the results compared to standard ns-3, and any difference could be explained simply by the random nature of the experiment.

Finally, we checked the packet and event traces from the simulations, searching for the discrepancies. We noticed that when only one connection is active in the link there are no packets dropped due to the congested link and we find only small differences among the simulators. However, when several connections are multiplexed simultaneously on the link and packets are dropped due to switch buffer overflow, a slight difference in transmission time computation or link delay results is not exactly the same packet being dropped by the switch. When a different packet is dropped in one simulator compared to another, it is probably from a different TCP connection. The result is that a different connection suffers an extended duration and the comparison of durations shows this difference.

We illustrate this situation in Fig. 8. In the left side of the figure, the result from a simulator is represented, where connection 3 suffers the packet loss, therefore it suffers an extended connection duration. The right side of the figure represents the output from a different simulator for the same connection arrivals, where the connection suffering the packet drop is connection number 2. When the connection durations are compared between both simulations, connections 2 and 3 present significant differences in duration while connection 1 does not.

We may compare the simulation results from Fig. 6 with those from the experimental setup described in Section II-B. If we compare the connection durations obtained from simulation with those from the experimental setup using the same traffic intensity (6 cnx/min/node), we obtain an average discrepancy of 233% for the ns-3 simulation, 260% for the OMNeT++ simulation, and 1397% for the ns-2 simulation. These large discrepancies may be the result of an incorrect switch buffer size configured in the simulations (50 packets). The actual buffer size in the circuitry of the physical Ethernet switch may be different, therefore it is not correctly modeled by a small 50-packets-deep output buffer.

We ran simulations using different buffer depths in the switching elements. Fig. 9 shows the average discrepancy between the simulations and the experiments for the same traffic intensity as in the results shown in Fig. 6 but varying the buffer size. We observe that above a certain buffer size the simulations do not change their results; they reach a behavior without any packet loss due to buffer overruns, so there is no difference when configuring even larger buffers. The stability point is reached for a different buffer size in case of ns-2 simulations compared to ns-3 and OMNeT++, proving different simulator behavior.

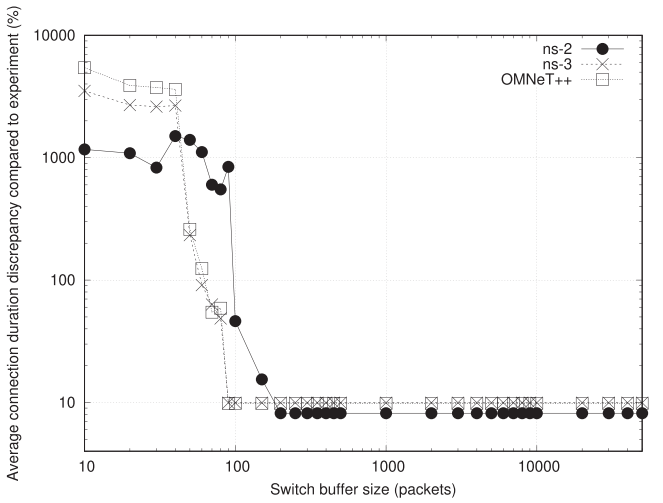

Fig. 9. Simulations varying switch buffer size compared with experimental results. Traffic intensity of 6 cnx/min/node.

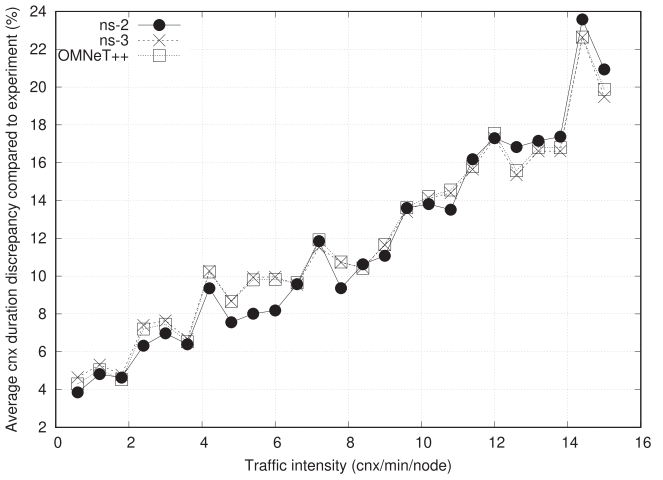

Fig. 10. Simulations varying traffic intensity for a simulated switch buffer size of 1000 packets. Comparison with experiments.

Both ns-3 and OMNeT++ offer very similar results although they differ for very small buffer sizes, when packet losses are extremely frequent.

The smallest difference between simulations and the experiments is obtained for large buffer sizes, when packet losses are less frequent. We configured a very large buffer size for the next simulations (1000 packets), which guarantees no losses and we obtained the average connection duration discrepancy for different traffic intensities. Fig. 10 shows the results for the three simulators compared with the experiments. For small traffic intensities, the discrepancy between simulations and the ground truth is small, close to 5%, however, it grows with traffic intensity up to more than an average 20% discrepancy. There is no simulator that consistently provides more accurate results with different traffic intensities, and ns-3 and OMNeT++ results keep being similar.

We could not obtain the actual maximum buffer size from the device specs, but only details about its drop-tail policy.

The circuitry in some Ethernet switches can allocate separate buffers for packets with different size, and memory for switching buffers can also be allocated dynamically, based on previous buffer overruns. It is therefore difficult to accurately simulate a real switch using a simple output-buffer queue and further implementation details from the manufacturer are needed.

We extended the simulations to scenarios modeling a long-distance inter-switch link (5 ms delay), different link bandwidth, different distribution of file transfer sizes or different link layer encapsulations, reaching similar conclusions. In any scenario subjective to chaotic behavior, a small difference in the simulation results in large differences when certain output values are evaluated.

IV. Conclusion

In this article, the predictive performance of different network simulation software packages was evaluated. The presented scenarios were applicable for the deployment of IoT devices in industrial communication networks, including data center or production plant deployments, large-distance links to collector servers, or large numbers of devices that send data in a nearly synchronized manner.

Three very frequently used simulators have been compared, i.e., ns-2, ns-3, and OMNeT++, which all entail independent implementations. We tested their networking stack by implementing congested links such that small changes in simulator implementation could amplify the differences for the metrics under study.

We have confirmed that small differences between the simulators (e.g., the difference of a single packet in each connection) lead to predictive errors of at least 15% when one simulator was compared to another that has been assumed to be the ground truth. We have also compared the simulation results with experimental ground truth by reproducing in simulation the results from experiments previously published or by implementing experiments with real equipment to recreate simulation scenarios. We have proven that some differences between simulators arise from small differences in the network protocol stack implementation. The chaotic behavior of congested links resulted in the amplification of the different results obtained from different simulators.

The different results provided by the simulators were not the consequence of an incorrect implementation in any of them but of the descriptions of many networking protocols which contain a large number of parameters and behaviors which are open to the implementers to specify. We could suggest cross-validation between network simulators; however, the large number of parameters existing in the protocols involved in a realistic TCP/IP communication over local or remote links make this unfeasible. The chaotic nature of congested systems made the perfect validation of simulators a task that require a nearly identical programming code, which invalidated any comparison. We prefer to emphasize this fact to the research community and practitioners of network performance evaluation. Just as we accept that evaluation testbeds were not generally realistic scenarios, we must also consider that the difficulty to capture and specify all the required protocol parameters generated error in the simulated results.

References

[1] R. Bush and D. Meyer, "Some internet architectural guidelines and philosophy," RFC 3439, pp. 1–28, Dec. 2002, doi: 10.17487/RFC3439.

[2] K. Park and W. Willinger, *The Internet as a Large-Scale Complex System*. Oxford, U.K.: Oxford Univ. Press, 2005.

[3] Z. Zhu and C. Fu, "A chaotic dynamical model for internet traffic," in *Proc. 4rth Int. Conf. Nat. Comput.*, 2008, pp. 581–585.

[4] C. Systems, "Best practices in core network capacity planning," 2013, Accessed: Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/routers/wan-automation-engine/white_paper_c11-728551.html

[5] Riverbed, "Riverbed modeler," Accessed: Aug. 2020. [Online]. Available: https://cms-api.riverbed.com/portal/community_home

[6] Simulcraft, "OMNEST," Accessed: Aug. 2020. [Online]. Available: https://omnest.com

[7] S. N. Technologies, "Qualnet—Network simulation software," Accessed: Aug. 2020. [Online]. Available: https://www.scalable-networks.com/products/qualnet-network-simulation-software-tool/

[8] TETCOS, "NetSim–Network simulator & emulator," Accessed: Aug. 2020. [Online]. Available: https://tetcos.com

[9] O. Ltd., "OMNeT discrete event simulator," Accessed: Aug. 2020. [Online]. Available: https://omnetpp.org

[10] "ns-2," Accessed: Aug. 2020. [Online]. Available: http://nsnam.sourceforge.net/wiki/index.php/Main_Page

[11] "ns-3 network simulator," Accessed: Aug. 2020. [Online]. Available: https://www.nsnam.org

[12] "JSim," Accessed: Aug. 2020. [Online]. Available: https://www.physiome.org/jsim/

[13] B. Moussa, M. Kassouf, R. Hadjidj, M. Debbabi, and C. Assi, "An extension to the precision time protocol (PTP) to enable the detection of cyber attacks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 18–27, Jan. 2020.

[14] G. Premsankar, B. Ghaddar, M. Slabicki, and M. Di Francesco, "Optimal configuration of LoRa networks in smart cities," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7243–7254, Dec. 2020.

[15] Z. Ahmad and M. H. Durad, "Development of SCADA simulator using Omnet," in *Proc. 16th Int. Bhurban Conf. Appl. Sci. Technol.*, 2019, pp. 676–680.

[16] P. Sousa and L. Ferreira, "Hybrid wired/wireless profibus architectures: Performance study based on simulation models," *EURASIP J. Wireless Commun. Netw.*, vol. 2010, 2010, Art. no. 21.

[17] B. Kim, D. Lee, and T. Choi, "Performance evaluation for modbus/tcp using network simulator ns3," in *Proc. TENCON IEEE Region 10 Conf.*, 2015, pp. 1–5.

[18] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of Things (IoT): Research, simulators, and testbeds," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1637–1647, Jun. 2018.

[19] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A library for parallel simulation of large-scale wireless networks," in *Proc. 12th Workshop Parallel Distrib. Simul.*, 1998, pp. 154–161.

[20] "Veins the open source vehicular network simulation framework," Accessed: Aug. 2020. [Online]. Available: http://veins.car2x.org

[21] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras, "ATEMU: A fine-grained sensor network simulator," in *Proc. 1st Annu. IEEE Commun. Soc. Conf. Sensor Ad Hoc Commun. Netw.*, 2004, pp. 145–152.

[22] L. Kanaris, C. Sergiou, A. Kokkinis, A. Pafitis, N. Antoniou, and S. Stavrou, "On the realistic radio and network planning of IoT sensor networks," *Sensors*, vol. 19, no. 15, Jul. 2019, Art. no. 3264. [Online]. Available: http://dx.doi.org/10.3390/s19153264

[23] Y. Wang, Z. Gao, W. Ji, H. Zhang, and D. Qing, "Exploiting task-based parallelism for parallel discrete event simulation," in *Proc. 26th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process.*, 2018, pp. 562–566.

[24] K. Pawlikowski, "Steady-state simulation of queueing processes: Survey of problems and solutions," *ACM Comput. Surv.*, vol. 22, no. 2, pp. 123–170, Jun. 1990. [Online]. Available: https://doi.org/10.1145/78919.78921

[25] A. M. Uhrmacher, S. Brailsford, J. Liu, M. Rabe, and A. Tolk, "Panel—Reproducible research in discrete event simulation—A must or rather a maybe?," in *Proc. Winter Simul. Conf.*, 2016, pp. 1301–1315.
[26] S. J. E. Taylor, T. Eldabi, T. Monks, M. Rabe, and A. M. Uhrmacher, "Crisis, what crisis—Does reproducibility in modeling simulation really matter?," in *Proc. Winter Simul. Conf.*, 2018, pp. 749–762.
[27] C. Gomez, A. Arcia-Moret, and J. Crowcroft, "TCP in the Internet of Things: From ostracism to prominence," *IEEE Internet Comput.*, vol. 22, no. 1, pp. 29–41, Jan./Feb. 2018.
[28] G. Ji, Z. Wang, D. Zhu, D. Makrakis, and H. T. Mouftah, "Performance evaluation and improvement of TCP/IPV6 over ieee 802.15.4 under Wi-Fi interference," in *Proc. IEEE 28th Can. Conf. Elect. Comput. Eng.*, 2015, pp. 1095–1100.
[29] S. R. Pokhrel and S. Singh, "Compound-TCP performance for industry 4.0 WiFi: A cognitive federated learning approach," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2143–2151, Mar. 2021.
[30] L. Guo and J. Y. B. Lee, "On TCP simulation fidelity in ns-2," in *Proc. 14th ACM Int. Symp. QoS Secur. Wireless Mobile Netw.*, 2018, pp. 55–62. [Online]. Available: https://doi.org/10.1145/3267129.3267132
[31] M. Bateman and S. Bhatti, "TCP testing: How well does ns2 match reality?," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2010, pp. 276–284.
[32] M. Casoni, C. A. Grazia, M. Klapez, and N. Patriciello, "Implementation and validation of TCP options and congestion control algorithms for ns-3," in *Proc. Workshop Ns-3*, 2015, pp. 112–119. [Online]. Available: https://doi.org/10.1145/2756509.2756518
[33] Y. Lu, Z. Ling, S. Zhu, and L. Tang, "SDTCP: Towards datacenter TCP congestion control with SDN for IoT applications," *Sensors*, vol. 17, no. 12, Jan. 2017, Art. no. 109. [Online]. Available: http://dx.doi.org/10.3390/s17010109
[34] G. Jereczek, G. L. Miotto, and D. Malone, "Analogues between tuning TCP for data acquisition and datacenter networks," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 6062–6067.
[35] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 63–74. [Online]. Available: https://doi.org/10.1145/1851182.1851192
[36] Y. Chen, R. Griffith, D. Zats, A. D. Joseph, and R. Katz, "Understanding TCP incast and its implications for big data workloads," *The USENIX Mag.*, vol. 37, no. 3, pp. 24–38, Jun. 2012. [Online]. Available: https://www.usenix.org/publications/login/june-2012/understanding-tcp-incast
[37] Y. Qin, W. Yang, Y. Ye, and Y. Shi, "Analysis for TCP in data center networks: Outcast and incast," *J. Netw. Comput. Appl.*, vol. 68, pp. 140–150, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516300649
[38] V. Vasudevan *et al.*, "Safe and effective fine-grained TCP retransmissions for datacenter communication," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 303–314. [Online]. Available: https://doi.org/10.1145/1592568.1592604
[39] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements and analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas.*, 2009, pp. 202–208. [Online]. Available: https://doi.org/10.1145/1644893.1644918
[40] R. Nishtala *et al.*, "Scaling memcache at Facebook," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 385–398. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/nishtala
[41] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast congestion control for TCP in data-center networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 345–358, Apr. 2013.
[42] M. Seewald, "Building an architecture based on IP-Multicast for large phasor measurement unit (PMU) networks," in *Proc. IEEE PES Innov. Smart Grid Technol. Conf.*, 2013, pp. 1–5.
[43] *BROADCOM*, "Broadcom high-capacity strataxgs trident II ethernet switch series," 2002. Accessed: Mar. 2020. [Online]. Available: https://www.broadcom.com/products/ethernet-connectivity/switch-fabric/bcm56850

Daniel Morato (Member, IEEE) received the M.Sc. and Ph.D. degrees in telecommunication engineering from the Public University of Navarre, Pamplona, Spain, in 1997 and 2001, respectively.

He was a Visiting Postdoctoral Fellow with the Electrical Engineering and Computer Sciences Department, University of California, Berkeley, CA, USA, in 2002. He joined the Department of Automatics and Computing, Public University of Navarre as an Associate Professor in 2006. He is currently an Associate Professor with the Department of Electrical, Electronic and Communications Engineering, Public University of Navarre. He also became a Member of the Institute of Smart Cities (UPNA), in 2014. His research interests include high-speed networks, performance and traffic analysis of Internet services and network monitoring.

Carlos Pérez-Gómara was born in Pamplona, Spain, in 1993. He received the B.S. degree in telecommunication engineering from the Public University of Navarre, Pamplona, Spain, in 2015.

He has been a Research Assistant for two years with the Telecommunications, Networks and Services Research Group, working in network simulation for data center, IoT, and wide area network environments. He has also collaborated in several research projects between the Public University of Navarre and the industry.

Eduardo Magaña (Member, IEEE) received the M.Sc. and Ph.D. degrees in telecommunications engineering from the Public University of Navarra, Pamplona, Spain, in 1998 and 2001, respectively.

Since 2005, he has an Associate Professor with the Public University of Navarra, in the area of telematics engineering and telecommunications. He was a Postdoctoral Visiting Research Fellow with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA, in 2002. He became a founding Researcher of the Institute of Smart Cities, Spain, in 2014. He is the Vice Principal with the Department of Electrical, Electronic and Communications Engineering, Public University of Navarre. His main research interests include network monitoring, traffic analysis and performance evaluation of communication networks.

Mikel Izal (Member, IEEE) received the M.Sc. and Ph.D. degrees in telecommunication engineering from the Public University of Navarre, Pamplona, Spain, in 1997 and 2002, respectively.

He was a Scientific Visitant with Institute Eurecom, Sophia-Antipolis, France, performing measures in network tomography and peer-to-peer systems, in 2003. He is currently an Associate Professor with the Department of Electrical, Electronic and Communications Engineering, Universidad Pública de Navarra, Pamplona, Spain. He is a Member with the Instituto of Smart Cities. His research interests include traffic analysis, network performance and monitoring, high speed next generation networks, and services and peer to peer systems.