International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Testbeds And Simulation Tools for SIoT Strategies To Shorten Time-To-Market For SIoT Solutions.

Venkateswara Raju K, Manjula R Bharamagoudr

Abstract—The integration of social networking and Internet of Things (IoT) concepts has attracted a growing interest of researchers and has promised to support new and powerful applications. However, the time it takes from development to the market for SIoT solutions is longer, especially if proper simulations are made to troubleshoot challenges in a timely manner. We propose the use of a multi-thread simulation tool that combines the existing testbeds, such as Network Simulator (Version 2), Devices Profile for Web Services, Objective Modular Network Testbed in Cplus plus, OPTNET, Parallel and Distributed Simulation (PADS), and J-Simtorun simulations in multiple threads to improve performance and save on time when simulating complex SIoT devices that are able to communicate with humans as well as interact with other devices. Through this approach, developers will shorten time-to-market for SIoT solutions and thus afford these solutions to users and improve the quality of life.

Keywords—SIoT; IoT; Test beds; Multi-level simulation

I. INTRODUCTION

A new era of communication is emerging where a variety of real gadgets such as RFID tags, sensors, and electromechanical devices that are used on a daily basis will all be connected to the Internet [1]. These gadgets can be integrated and become human assistants that are able to carry out daily tasks in a natural and seamless manner, using intelligence and information derived from the network that connects them [2]. Such a pervasive paradigm of Internet of Things (IoT) has a potential to increase information value that is generated by different interconnections between gadgets to gadgets and gadgets to people. This translates a transformation of information into beneficial knowledge that can improve the life standards of people and the society [3]. The inherent challenges faced by people and organizations could be addressed with the introduction of IoT applications and smart services which facilitate the connection of objects to objects or objects to humans at any time and any place [4].

Manuscript published on 30 September 2019
* Correspondence Author
Venkateswara Raju K*, School of Electronics of Communication Engineering, REVA University,
Bangalore, India {E-mail: kvrajukonduru@gmail.com}
Manjula R Bharamagoudr School of Electronics of Communication Engineering, REVA University,
Bangalore, India {E-mail[2]: manjula@revainstitution.org}

The vision of IoT to pervasively connect billions of objects ushers an era where such objects are able to interact with the natural environment and receive information regarding their statuses [1].

While local networks, such as wireless sensor networks (WSN), smart homes, and machine-to-machine communications allow the extraction of regional information with specific content, IoT offers a comprehensive, large-scale, and historical information through the collaboration between diverse intranets of things regardless of the heterogeneity of local communication technologies, devices, and goals of deployment [5].

IoT is also capable of creating new applications and services, thus providing a new ecosystem where a number of objects can interact and collaborate [6].

Humans often interact with other humans in different ways in their lives. While interacting, humans use a variety of applications and services to improve their quality of life. Therefore, the quality of experience (QoE) of smart applications and services will largely depend on how they satisfy the needs arising from human relationships. Equally, the high degree of correctness of the needs is largely derived through the collaboration of humans in relation to the quality of service. Through IoT, individual users of services and applications connect to others through a legacy network. At the same time, a number of things interact with each other through the internet, which offers information to smart applications and services as each of the users use them [20]. IoT can therefore be said to interact in two different paradigms: human-to-human paradigm and thing-to-thing paradigm. This implies that IoT fails to establish a true connection between things and humans for effective ubiquitous computing. The concept of social Internet of Things (SIoT) was adopted and advanced to practically integrate pervasive computing with high QoE into daily lives. In order to achieve such an objective, the connectivity of the relationships between things and humans must be enhanced.

The deliberation of socially networking humans to things for pervasive computing has been seen to go beyond IoT. The SIoT concept facilitates the socialization of things so that humans can establish relationships with them in a seamless manner [2]. Not only are the connections physical, but also logical connections of social communities that involve things and humans. However, while making attempts to improve such interactions, challenges such as privacy, security,

resource-constraints, and the time it takes to have SIoT products on shelves have become a concern for academia, industry, and users. To reduce the time-to-market of SIoT products, proper simulation should be done to ensure that the products work as expected.

This paper explores examples of test beds and simulation techniques such as NS (Version 2), DPWS, IOTIFY, OMNeT++/NET, Windriver, NetSim, PADS, and ASSIST that could be used to ensure that SIoT paradigms work as intended by characterizing the initial conditions and execution parameters, address the issue of resources constrained environments of physical hardware, and shorten time-to-market of SIoT solutions.

II. SIOT CHALLENGES THAT PROLONG TIME-TO-MARKET

It is estimated that there will be about 26 billion IoT devices by 2020, and according to ABI Research, about 30 billion devices will be connected to the IoT wirelessly by the same year. In the market today, there are a number of devices that can be remotely accessed and controlled using personal computers and smart phones. These devices often work in different ecosystems that differ in terms of domains, regions, and applications. Different social networking platforms connect people sharing common interests or activities. In 2014, Pew Research found that about 74 percent of online adults shared through social networking platforms. Through the growing ubiquity of computing, people are increasingly generating and sharing content with other people via social networks. SIoT connects the IoT with these social networking platforms. This is a novel paradigm that provides an ecosystem which allows smart devices to easily interact with people within a social network [3].SIoT allows seamless collaboration and connection of users and things through the social networks. Things in SIoT are active participants performing social roles, and the knowledge and information from the devices gets shared with other devices and humans within the network. In the future, humans will become more cooperative in SIoT and will cooperate toward common goals. However, there are inherent problems relating to the question of developing SIoT solutions and reaching the end user within an appropriate duration.

The first challenge is that the existing SIoT solutions are insufficient to achieve the interoperability of the varied services and devices that are manufactured by different vendors throughout the world. Every manufacturer tends to offer their own APIs that are used in accessing such devices and this poses a challenge when it comes to interactions between devices and humans. It therefore emerges that the main challenge is for humans to utilize all the functionalities that are provided by different devices sharing similar capacities and to offer proper abstractions of the details during implementation.

The second challenge is that while devices are able to communicate with each other, no available mechanisms exist that will properly discover and share relevant and useful information in measure. For instance, in a case where multiple homes have heating systems, it is likely that neither the home owner nor the heating system knows it are able to save energy by opting for a more appropriate configuration for the parameters of the system [32]. However, if the entire heating systems were capable of sharing their configuration details with other SIoT objects in the social networks, then an individual object would be able to learn the configuration that would lead to a better and energy-efficient operation.

The other SIoT challenge is that there is no established methodology for how to apply the discovered information from humans and devices in order to achieve a common objective in a collaborative manner. For example, the existing home security systems often use dedicated devices which are managed by the owner of the home and the security company offering the service. However, within the SIoT vision, different devices in a user's home will be able to participate to achieve a common goal, which is to secure the user's home. Friends within the social network will also be able to participate in a dynamic manner to achieve a particular goal depending on their location. Home devices will be able to detect any intrusion in a collaborative manner by employing their own capacities based on the custom instructions executed for the users' homes. For instance, motion sensors in laptop video cameras or thermostats will be able to detect any peculiar motion and capture the scenes. Alarms will then be issued by both making sounds and turning on the TV while displaying a message. The alarm will be relayed to mobile phones within the range of the user's home.

The challenges outlined above largely prolong the time it takes for SIoT products to reach the market. Simulations and testbeds that can significantly address these challenges as products are first tested before they are available to the end users. Product designers that deploy simulation in their design process have been shown to outperform those that do not employ the tools in terms of meeting the cost of products, launching products in time, and meeting the quality and revenue targets as shown in Fig 1. Manufacturers who push simulation into the design process have enjoyed significant product metrics, saved product rework time, decreased development time, and reduced prototype development [4]

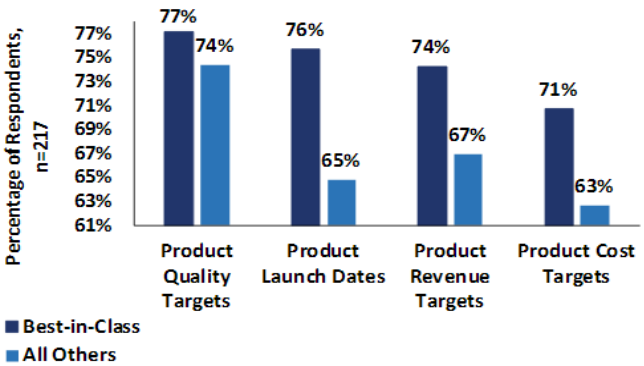


Fig 1. Simulation-driven design in meeting targets [4]

III. SIOT TESTBEDS AND SIMULATION TOOLS

a) NS (Version 2)

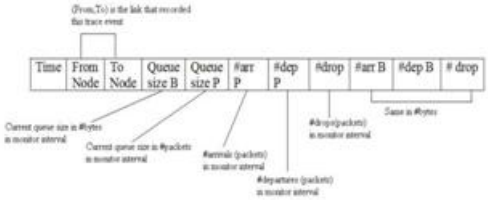
NS is an object-oriented and discrete-event-driven simulation tool first developed by researchers at UC Berkeley.

It is a network simulator that is written in Object-oriented Tool Command Language (OTcl) and C++. The tool is particularly useful for simulating a number of IP networks, including both wide and local area networks and can therefore be used to diagnose the efficiency of SIoT objects to connect to the Internet before they are released to the market.

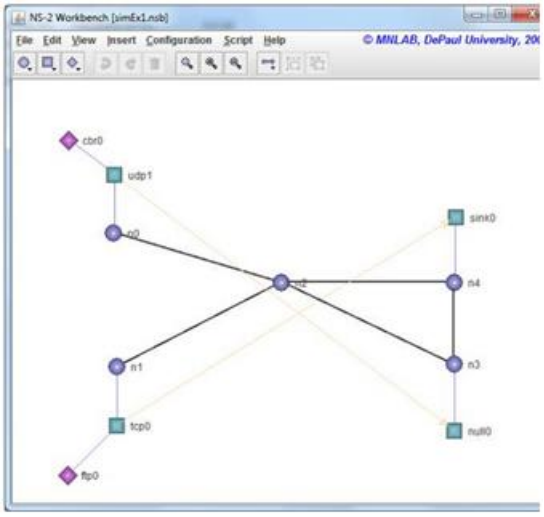
NS can be used to implement UPD and TCP, and check traffic source behavior such as VBR, CBR, FTP, Web, and Telnet. It also implements multicasting as well as some common types of MAC layer protocols typical to LAN simulations [5]. While NS tool is easy to use, it might be challenging for first-time users as the available manuals are not user-friendly. NS is generally an Object-oriented script interpreter with network component object libraries, simulation event scheduler, and network setup module libraries. NS is used by programing in OTcl language and setting up and running the simulator; users write OTcl scripts which initiate event schedulers [21]. OTcl is used to configure and create a network while C++ runs the simulation. In order to create an executable file, C++ codes must be compiled, and linked. NS2 runs on Windows platform using Cygwin and requires a memory of about 250MB. The tool can be downloaded from the Cygwin website and installed using the Wizard as shown in Fig 2.

Fig 2: Cygwin Wizard to download and install NS2 [5].

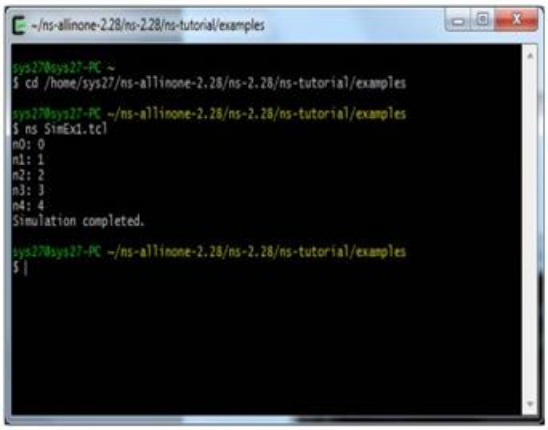
NS2 can be used for queue monitoring or tracking the packet dynamics at a queue or other SIoT objects. Queue monitoring allows for tracking packet arrival, departure or drops, and can be used to compute the averages of such values. Important queue information can be obtained using queue monitoring, as shown in Fig 3.



In the following example, NS workbench is employed in the generation of TCL scripts and the creation of scenarios that are run in NS2 to generate NAL files and trace files. In order to use NS workbench, systems must have Java Sdk installed first [31]. Ns-becnh jar file must be downloaded and executed. In the example shown below, the network has five nodes, beginning with n0 to n4. Node n0 is sending constant bit-rate (CBR) traffic to the fourth node n3, while n1 sends data to n4 using FTP as show in Fig 3. The two traffic sources are conveyed using UDP and TCP respectively. The transmitting objects of the two protocls in NS2 are UDP and TCP agents and the receivers are Null and TCP sink agents respectively.



The TCL script can then be generated and saved. In order to run the TCL script in NS2, Cygwin should be started and the directory path to the folder the TCL script in saved to be changed. NS command is used to to run the TCL script as shown in Fig 4 below.



After the simulation, two files will be generated: OutEx1.tr and OutEx1.nam in the same folder. NS2 is recommended for testing SIoT objects as it has a number of advantages over other simulators in the market. NS2 is cheap as it does not necessarilly require expensive equipment. In complex situations, NS2 can be tested with each and the results are obtained faster. The tool is also popular in the industry and academia.

b) Devices Profile for Web Services (DPWS)

Since its infancy, the evolution of IoT to SIoT has seen the introduction of applications and services in low-power wireless computing and communication paradigms in resource-constrained settings [6]. One of the challenges within these environments is to integrate social networking services into smart objects for a seamless integration of the functionalities into the Web.

Addressing this challenge would mean that there will be potential to have new SIoT applications where several devices can be connected with a number of existing Web services and resources. Devices Profile for Web Services (DPWS) is a powerful simulation tool that can help realize this objective. The DPWS simulation toolkit allows for a secure connection of resource-constrained objects to the Web. Its architectural concept is similar to that of Web Service, although it differs in a number of ways making it appropriate in resource-constrained settings and event-driven cases. DPWS is largely based on Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) and can define and communicate service in different objects although it does not need central service registry in order to discover services. Instead, DPWS uses UDO multicast and SOAP-over-UDP binding in dynamically discovering device services. It offers a subscribe/publish eventing mechanism for device-event client subscription. For instance, such device events may include switches of the devices being on or off or temperatures attaining a predefined threshold. When any event occurs, the notifications are relayed to subscribing objects through other TCP connections. Most European projects that are supported by the Information Technology for European Advancement have used DPWS as an important technology for solving a plethora of technical issues with smart objects. As more objects and services get into the market, DPWS has been undergoing constant implementations for a number of resource-constrained objects and companies are increasingly adopting DPWS technology in testing and simulating applications and services before launching products in the market. While DPWS first made its debut in 2004, there are a few tools that have been developed which support DPWS applications. Han and his group have developed a simulation toolkit known as DPWSim that helps developers to make prototypes, develop, and simulate their applications before marketing. This tool can significantly reduce the time-to-market for SIoT solutions and improve the quality of such products. DPWSim can mimic all the DPWS protocols and software features in an intuitive interface in order to offer an efficient manner to manage and simulate SIoT devices. The main advantages of DPWSim include intuitive GUI, ability to generate virtual devices, platform independent, flexibility, and requires no change of code for SIoT applications following simulation [6].

DPWSim supports the development of SIoT solutions using DPWS by creating virtual devices that are discoverable on a network and can interact with other clients of objects through the DPWS protocols. In addition, the toolkit is capable of simulating environments where DPWS devices work in as well as create and manage simulations before storing and loading them. There are four components of DPWSim: spaces, devices, operations, and events, as shown in Fig 5 below. Spaces are virtual environments that represent the real-life environment where SIoT devices operate in. this can be an office, a home, a public space, or a train station. Devices are the hosted and the hosting DPWS services and operations are different processes within a device that reflect its functionalities. Events are similar to operations which ensure the implementation of a given functionality.

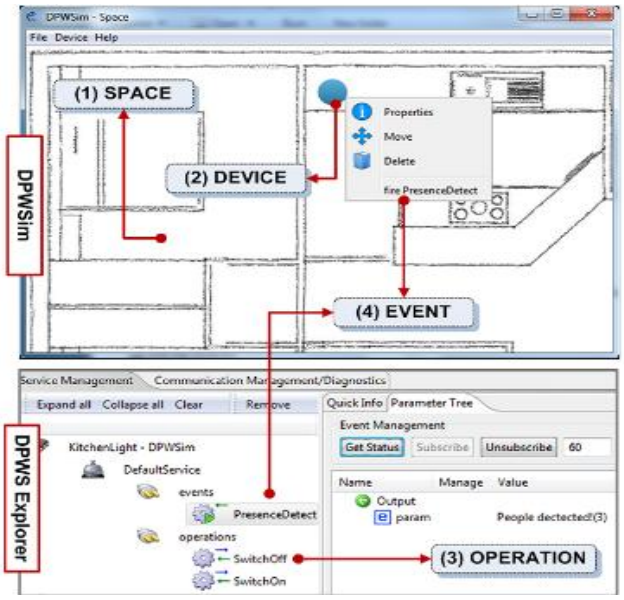
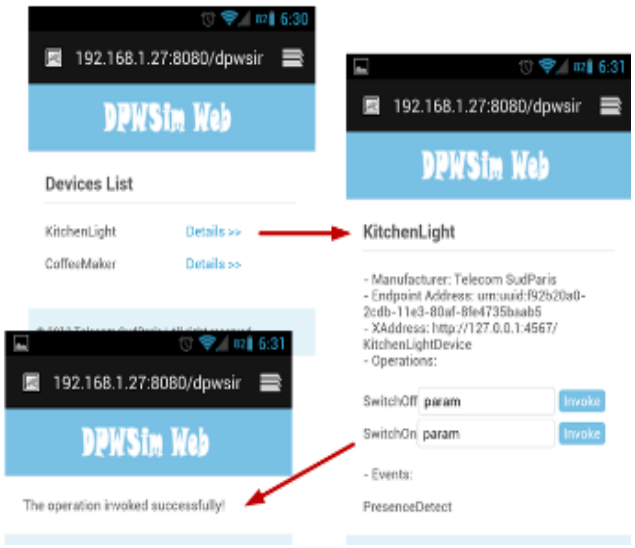


Fig 5 Compoents of DPWSim [6]

DPWSim has been used in DPWS Explore, the standard tool used among the DPWS community. DPWS Explorer is mainly a tool for analyzing the compliance of DPWS services. The toolkit can visualze a number of aspects of both hosted and hosting services as message exchange and metadata and provide the capacity to subscribe to a service or call events and operations. It is the common tool used to preview DPWS services in the process of IoT development. The tool discovers all the virtual devices as well as invoke their operations and the virtual devices' events can be subscribed right from the toolkit as shown in Fig 6. As shown in Fig 6, users can opt to turn on KitechnLight light bulb through the invocation of the operation SwithOn throgh their smartphone interfaces of the toolkit DPSWSin Web.



c) OPNET

OPNET refer to a collection of simulation tools that are based on a similar simulation engine. This toolkit is manufactured and shipped by OPNET Technologies. Originally, OPNET was developed in 1987 at the Massachusetts Institute of Technology (MIT) and today the simulation tool is commercialized but not widely available. The simulator comes with full source code for simulation modules, exhaustive documentation, and examples [35].

Additional support is also available when users subscribe to mailing lists and the simulation tool requires maintenance license. OPNET is written in C, although recent versions support C++. The initial parameter setting and topology setup is achieved using GUI, which are sets of XML files, or via C library calls. The simulation scenarios require users to write C or C++ codes, although in simpler incidents special scenario parameters can be used. However, implementation of OPNET modules has been shown to be complex and largely fragile compared to ns-2. Modifying OPNET behavior is also difficult, requires more time and effort and is always error-prone [34]. One common feature of OPNET is that it has detailed prototypes of network equipment and the details reflect the real network behavior. This feature makes it effective in simulating SIoT devices, although the toolkit has its own demerits, such as hanging up while simulation process [9]. In some cases, packet processing in this toolkit mirrors closely how network equipment works in real time. For instance, routers in OPTNET will receive a packet, and, if necessary, decompress it, and select the outgoing interface in case the packet is not addressed to the device. The router will then transfer the packet to the buffer of the outgoing packet. The processing and packet decompression times must be considered when selecting OPNET as a simulation tool.

Simulation in OPNET is largely based on a discrete event queue and early versions of OPTNET were single threaded and synchronous. In such versions, small portions of code could be parallelized on a number of processors. Today, OPTNET supports multithreading and parallel execution on multi-processors although each processor requires its own license. OPNET supports a parallel execution window that defines time intervals where parallel event execution from various instances of modules is allowed. Because the network equipment models in OPTNET are largely detailed and since the simulation processes shows what happens in real-world in a network device, simulating with OPTNET tends to be relatively time consuming and requires thread scheduling protocols of the operating system. OPTNET is also equipped with a GUI interface, which is advanced, although one can launch the simulator from a command line, and the default mode for OPTNET is the GUI [9]. The OPTNET's lack of a scripting language has been found to be a limiting factor among developers.

d) J-SIM

J-SIM or JavaSim (as formerly named) was developed by researchers at Distributed Realtime Computing Laboratory (DRCL) at Ohio State University. Other third-party packages can also be downloaded. Unlike OPTNET, J-Sim is freely available and comes with source code, white papers, and tutorials. A mailing list is also available for public access. The tool is implemented in Java and Tcl in the form of Jacl. While Java is used in the creation of objects in J-Sim, known as components, Tcl allows for topology setup and offers a limited way of simulation control. Just like in ns-2, J-Sim provides a high-level interface that is based in Tcl programming. Earlier versions of J-Sim did not support the building of model extensions. The simulator is based on Autonomous Component Architecture (ACA) and its components are all written in Java programming and expose clear interfaces. These components often interact with each other via ports and are mapped to contracts. The contracts determine component interactions and permit the use of services of the lower layer, although they also introduce strict coupling between the layers [25]. Components are often assembled by wiring the ports at runtime and are organized in the form of a tree-like model where a parent component is always a directory for each child. J-Sim's Runtime Virtual System (RUV) refers to Tcl commands that are used to manipulate the tree components and the hierarchical names of the component are similar to file paths.

e) Multi-thread simulation tools

The growing number of IoT objects and their connections to the social network has demanded novel simulation techniques to improve scalability and enhance real-time execution in largely populated SIoT environments, such as large-scale smart cities [7]. At the same time, the heterogeneous scenarios have necessitated manufacturers to resort to using complex simulation and modeling techniques to avoid wastage of time in the production of SIoT devices. A two-level simulator is proposed to work effectively in such environments. Two-level simulation is largely based on adaptive parallel, coarse-agent, and distributed-based simulation to create a model of the simulated objects' general life [7]. In some scenarios, a finer grained simulator, often based on OMNet++, can be triggered on some sections of the simulated area to allow the consideration of issues that arise when connecting objects using wireless network. Therefore, ad-hoc wireless networking does work effectively in the deployment of smart objects over a decentralized countryside. At the same time, the performance evaluation shows that the employment of multi-level simulation is a viable solution when simulating SIoT environments in scale. Multi-level simulation tools are capable of simulating the scenario in the smart market in shire environments. Scalability is the major requirement in terms of granularity of events and the modeled entities while using this approach. In such environment, even the smallest smart shire has thousands of interconnected objects and a majority of them are often mobile with each device portraying unique technical and behavioral characteristics [27]. Apart from scalability, simulators need to run in real-time, allowing a proactive approach and performing "what-if analysis" when managing the set architecture. The approach that serves the above requirements is the multi-level approach which combines both agent-based smart-shire simulator and discrete-event simulating engine.

The multi-level simulator has two levels of simulation: the coarse level denoted as Level 0 and the OMNeT ++ simulation known as Level 1. Level 0 simulation works to model the entire smart territory where a number of actors can subscribe their unique interests, produce products, and move to varying geographical locations. This can be implemented by employing an agent-based simulator fitted with both distributed and parallel execution potentials. Specific interactions in the smart market necessitate more simulations detail in the consideration of wireless network issues, movements and fine-grained interactions.

In order to address such issues, the implementation of Level 1 simulation, such as OMNeT++ simulation instance, is recommended. The main issue in this simulation is to ensure that both simulators are interacting. Level 0 will trigger Level 1 when required and will pass arguments to Level 1, which would function as initialization and configuration parameters as shown in Fig 6. Level 1 simulator will need to run for a specified amount of timesteps and as it approaches the end, it must notify Level 0 about its own outputs. Level 0 will also request Level 1 to carry on with its simulation or choose to end the simulation. In level 0 simulation, an agent-based simulator such as Smart Shire Simulator (S^3) can be used. Based on GAIA/ARTIS simulation middleware, the S^3 simulator performs parallel/sequential/distributed simulation in large scale by using a number of synchronization methods and communication approaches. In Level 1 simulation, OMNeT++ can be used with INET framework to simulate fixed nodes [7].

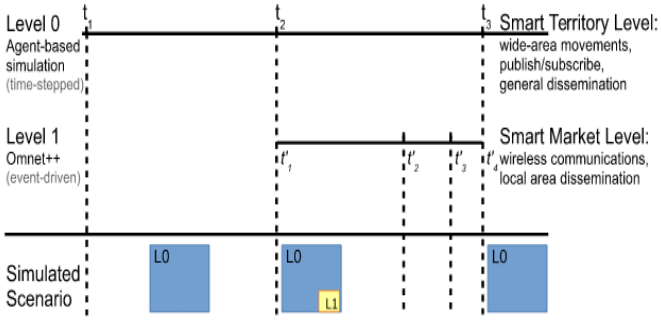


Fig 5. Multi-level simulation [8]

IV. MULTI-THREAD SIMULATION CONCEPTS

The need to quickly analyze network protocols or SIoT infrastructure has shaped the evolution of multi-thread simulations that provide real-time scale simulations. Multi-core systems are widely available today and performing multiple threading parallelisms has been made simple. Such a concept has already been implemented in NS3, a replacement of NS2 to improve its performance [10]. Other researchers have implemented threaded simulation in creating tools such as A Simple Simulator based on Threads (ASSIST) [11]. Implementation of parallelism for NS3 aimed at both speeding up the simulation process and making the toolkit as less intrusive as possible for the users. Multithread parallelism is less intrusive compared to distributed parallelism as only a single computer is used. In distributed parallelism, the distributed algorithms will need static partitioning simulation of the network [10]. Elsewhere, time-step simulation models have been recommended for prototyping solutions to prolonged time taken to develop systems using multiple cores on single dies [12]. While new designs of microprocessors have incorporated multicores, simulation technology has lagged behind, resulting in prolonged time spent on simulations in the design cycle. Researchers now find the use of uniprocessor simulation tools cannot be used for multiple core simulation [12]. For complex and detailed simulations, it is recommended that multiple threads be used in simulating systems for faster results [13].

Parallel programming entails decomposing programs into discrete elements which can be executed in a simultaneous manner by tasks that employ multiple computation resources, including microprocessors that are connected to a network. A number of problems solved using sequential programing or algorithms can be decomposed into discrete parts in order to make parallel versions of the same programs. Many issues encountered in simulations have been seen to be solved using parallelization. While discrete-event network simulation is part of the solution, the identification of the optimal manner to break the simulation or decomposition is critical [14]. Parallel computing architectures are today available in the market in varying sizes and shapes. A Symmetric Multiprocessor (SMP) presents a case of Uniform Memory Access (UMA) with multiple numbers of identical cores or processors fabricated into one die or integrated circuit share a single memory space. Resident and cache memory are then accessed in a symmetric manner with each SMP core having uniform access time. Another form of shared memory architecture that combines a number of SMPs is Non-Uniform Memory Access (NUMA) [26]. In this architecture, SMPs have their individual local memory space that is mapped into global address space and shared by processors. For shared memory systems, programming is usually done in threads or individual paths of execution in every single process. Multiple SMP machines that are interconnected in a network are involved in distributed memory systems. In such systems, memory spaces are often local to every machine, and as a result, each task executed on separate computers has to communicate over the network in order to access the required remote resources. This model is known as message passing and has been employed by Remote Direct Memory Access (RDMA) and like technologies that permit direct access to remote computers or nodes without necessarily involving CPUs on remote hosts [28].

In distributed parallelism, processes that run concurrently on similar or different computing resources have no access to program variables and have to coordinate in order to run one program by overtly sending or receiving messages. This intra-node communication model involves ranks or processes in parallel programming communicating with other ranks on SMP-based nodes or processors. Inter-node communication on the other side involves the communication over network interfaces with the processes on remote computers. The latencies that are involved internode communication are greater compared to the former case because of the added delays that are inherent in hardware buses, networking hardware, and host-channel interfaces [33]. This aspect is common in domain decomposition because specific processing activities requiring a number of inter-process communications are far better if they are mapped to single nodes. The de facto standard used to program distributed systems is known as Message Passing Interface (MPI), where blocking and non-blocking routines are supported and communication can be done either synchronously or asynchronously.

V. MULTI-THREAD SIMULATIONS AND RELEVANCE IN REDUCING TIME-TO-MARKET

There are multiple numbers of nodes involved in SIoT, making the application of fine-grained simulation tools emerge with scalability challenges in terms of performance evaluation. This means that monolithic testbeds handling these multiple nodes in SIoT to implement the required fine-grained level of detail are unable to offer simulated results in the required time. The delays in simulation also prolong the time it takes for SIoT solutions to be available in the market. Even when using the parallel programming approach, the massive numbers of devices in settings such as smart cities makes it difficult to perform quality simulation. However, these challenges can always be countered by either employing execution platforms using High Performance Computing (HPC) or reducing the detail levels in simulation models [29]. The two alternatives to addressing these challenges are not feasible because HPC platforms are often costly and reducing simulation detail levels will lead to misleading results since excessive details will be taken off from the simulation [8]. Therefore, a multi-level simulation and modeling approach would be the most recommended approach for highly populated SIoT environments.

Multi-level simulation allows the combination of approaches into a single model, with each tool performing specific task and working at varying detail levels. Using this approach implies that the model employs adaptive PADS simulator which works at coarse-grain detail level and that ensures the execution of domain-specific low or medium-level simulators which are only used if fine grained detail level simulators such as OMNet++ are required. Multi-level simulators switch between fine-grained and coarse-grained models automatically or the events can be triggered using simulator modelers. For instance, in an area that is overpopulated with wireless devices, as in the case of smart cities, a detailed simulation model can be triggered to collect information regarding congestion or network capacity issues. However, multiple-level simulations might pose a challenge in terms of interoperability since different simulators with varying standards are involved. The other issue that is unique to this approach is the design of inter-model communication platforms, such as state exchange and synchronization at runtime between the simulation components.

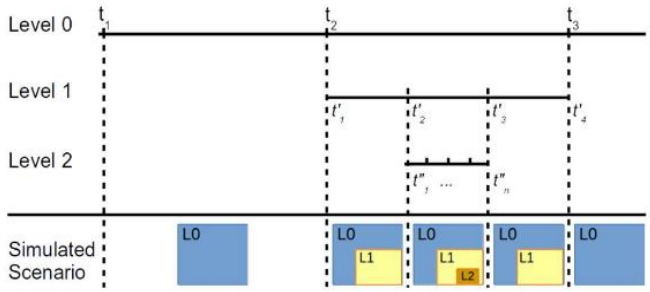


Fig 6 Simulation at multiple levels.

As shown in Fig 6 above, simulation results lack details at level 0 and the high-level simulator such as GALA/ARTIS) will manage the evolution of the simulation components as well as their communications after a time-stepped synchronization model. A section of the simulated scenario at timestep t_2 needs to be simulated with finer details, implying that in Fig 6 above some parts of the simulated area are at level 0 and some sections are modeled at level 1. If required, specific areas can be modeled to provide more details by employing the level 2 model. In general, considering two levels, it can be concluded that the entire simulation components that are managed by level 0 are thus evolved by t-sized timesteps. Contrary, components managed by all other levels employ *t'*-sized time steps [30]. Timestep t_2 refers to the time when some components of the model are transferred from course-grained to fine-grained simulation. At level 0, the components jump from *t2* to *t3* and all the others that are simulated at level 1 get updated at *t'2, t',3* and *t'4*. However, because there is no need for such detail, all the components that are simulated at level 1 are eventually transferred to the level 0 simulator. Because of the challenges resulting from the time-stepped synchronization model, interactions among simulated components at any course grained timestep, but all interactions at level 1 can take place at all fine-grained timesteps. Lastly, interactions between the components that are managed at various levels can take place only at coarse-grained timesteps. This implies that this happens if there is a match between timesteps.

By using multi-level simulation model, the number of nodes that can be handled by the toolkit never changes, although the level of details in the simulation evolution gets adapted to the simulation evolution need at runtime. This implies that the simulation model never runs at lowest levels of detail during the entire period of simulation. Therefore, it is feasible to have better scalability with multi-level simulation compared to monolithic approaches of simulation. However, it is also evident that multi-level simulation can introduce some error in each analysis [24]. This requires appropriate validation and verification techniques to be employed in each simulation. A practical example where multi-level simulation can be used is in smart cities and decentralized areas/smart shires. In smart shires, decentralized environments are capable of managing human, natural, buildings and infrastructural resources in a sustainable manner and a way that does not harm the environment. The idea of smart shires is to create smart, novel, and cost-effective services which can improve the lives of people [23]. The need for affordable solutions has spurred the adoption of crowd-sourced and crowd-sensed information that can be collected from people and SIoT interacting together. Today, sensors have become relatively low-priced and it is feasible to deploy them in any countryside. The sensors are connected using smart connection modalities. Devices with sensors are able to sense data and information processing technologies are used to manage and distribute important information to users. This creates a context-aware setting in smart shires [22]. Today, proximity-based social networking is possible due to SIoT. Users are able to see who is close to them using mobile devices connected to the Internet. Social network platforms such as Facebook are already using this model to let users know who is close to them. With the advancement in technology, more SIoT solutions will be created and improve the lives of people.

VI. CONCLUSION

This paper has explored SIoT challenges that prolong the time-to-market of devices and offered that testbeds and simulation tools could largely serve to reduce the time it takes for solutions to reach the end user. NS2, a simulation tool, has been used as an example of how modeling device connections could address challenges before launching SIoT devices in the market. In the paper other tools, including OPNET and J-Sim, have been discussed regarding the way they can be used to simulate and model smart objects before they can be introduced in the market. Other multithread simulation tools have been shown to work better in large-scale environments with thousands of devices such as in smart cities. The combination of coarse-grained and fine-grained simulation has been discussed and how such an approach can significantly achieve scalability. With reduced time of simulation, more SIoT solutions will be available in the market for consumers. The main goal of SIoT product designers is to avail their innovative products to the market as quickly as possible. In the past years, designers used to complete the initial stages of the design process then turn them over to dedicated groups of experts in simulations for optimization and iteration. While these groups have highly been regarded, they introduce further bottlenecks in the product design process. Simulation tools have offered solutions to such challenges in design process and have reduced the cost of product design, quality, and time-to-market. As a result, the use of simulation tools leads to more innovative products in the market and increases the chances of products more like to function at the first time without issues. Simulation tolls also reduce the likelihood of products requiring significant rework later in the design cycle.

REFERENCES

1. A.M. Ortiz., et al. The cluster between Internet of things and social networks: Review and research challenges. IEEE Internet of Things Journal, 1(2014)3:206-215
2. M. Nitti., M. Fadda., & M. Murroni. Exploiting Social Internet of Things features in cognitive radio. (December 2016) DOI: 10.1109/ACCESS.2016.2645979
3. L. Atzori., & D. Carboni. Smart things in the social loop: Paradigms, technologies, and potentials. Ad Hoc Networks, 18, (July 2014), 121-132
4. Internet Society. The Internet of Things: An overview. (October 2015). [Online] <https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151014_0.pdf> Accesssed 14th June, 2017
5. ENISA. Ad-hoc & sensor networking for M2M communications: Threats landscape and good practice. (January 2017). [Online] <> Accessed 14th June, 2017
6. A. Zanella. Internet of Things for smart cities. IEEE Internet of Things Journal, 1 (February 2014)1:22-32
7. L. Atzori., A. Iera, G. Morabito, & M. Nitti. The Social Internet of Things (SIoT) – When social networks meet the Internt of Things: Concept, architecture and network characterisation. Computer Networks, 56(November 2012)16:3594-3608
8. J.E. Kim., A. Marion., & D. Mosse. Socialite: A flexible framework for Social Internet of Things. 2015 [Online]<https://www.researchgate.net/publication/283142240_Socialite_A_Flexible_Framework_for_Social_Internet_of_Things. Accessed 6th June, 2016
9. Aberdeen Group. The benefits of simulation-driven design. 2017. [Online]<http://v1.aberdeen.com/launch/report/research_report/16443-RR-simulation-driven-design.asp> Accessed 8th June, 2017
10. J. Chung., & M. Claypool. NS by example. N.d [Online] <http://nile.wpi.edu/NS/ > Accessed 8th June, 2017
11. S.N. Han. DPWSin: A simulation toolkit for IoT applications using Devices Profile for Web Services. [Online]<https://pdfs.semanticscholar.org/5b89/97af4c70aba63317911eb4b8f7b0a159ed1a.pdf >Accessded 10th June, 2017
12. G. D'Angelo, S. Ferretti, & V. Ghini. Simulation of the Internet of Things. Proceedings of the IEEE 2016 International Conference on High Performance Computing and Simulation (HPCS 2016). September 2016 [Online]<https://pdfs.semanticscholar.org/f8be/13c0a971db4a73b1aa525d96dfc8e4569c95.pdf > Accesed 8th June, 2017.
13. G. D'Angelo, S. Ferretti, V. Ghini. Multi-level simulation of Internet of Things on smart territories: Simulation modeling practice and theory, Elsevier, vol. 73(April 2017) [Online]<https://arxiv.org/pdf/1611.01325.pdf>
14. M. Malowidzki. Network simulators: A developer's perspective. N.d. [Online] <https://pdfs.semanticscholar.org/24af/36a5f7ebc5f0bdfb643d5830c56278da2ff5.pdf > Accessed 13th June, 2017
15. G. Seguin. Multi-core parallelism for ns-3 simulator. August, 2009 [Online] <https://guillaume.segu.in/papers/ns3-multithreading.pdf> Accessed 12th June, 2017
16. K. Erciyes. Distributed graph algorithms for computer networks. London: Springer-Verlag (2013)
17. P.D. Bryan., J.A. Poovery, J.G. Beu,., & T.M. Conte. Accelerating multi-threaded applucation simulation through barrier-interval time-paralleism. (n.d) [Online]<http://tinker.cc.gatech.edu/pdfs/MASCOTS2012_Paul.pdf> Accessed 12th June, 2017
18. Reliasoft. Single-thread vs multi-thread. 2017. Online <http://www.reliasoft.com/BlockSim/multithread.htm> Accessed 12th June, 2017
19. R.G. Ford. A software testbed for simulation of cellular wireless netwoks. 2012. [Online] <http://wireless.engineering.nyu.edu/static-homepage/tech-reports/ford_msthesis.pdf > Accessed 13th June, 2017
20. D'angelo, M., Ferrari, A., Ogaard, O., Pinello, C., & Ulisse, A. 2012, February). A simulator based on QEMU and System C for robustness testing of a networked Linux-based fire detection and alarm system. In Proc. of the Conf. on ERTS2 (pp. 1-9).
21. Asensio, A., Castro, A., Velasco, L., & Comellas, J. 2013. An elastic networks OMNeT++-based simulator. In Transparent Optical Networks (ICTON), 2013 15th International Conference on (pp. 1-4). IEEE.
22. Gaias, G., Ardaens, J. S., & D'Amico, S. 2012. Formation Flying Testbed at DLR's German Space Operations Center (GSOC).
23. Vlachou, C., Herzen, J., & Thiran, P. 2014. Simulator and experimental framework for the MAC of power-line communications (No. EPFL-REPORT-205770). EPFL.
24. Lucas, J., Lal, S., Andersch, M., Alvarez-Mesa, M., & Juurlink, B. 2013. How a single chip causes massive power bills GPUSimPow: A GPGPU power simulator. In Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on (pp. 97-106). IEEE.
25. Radak, J., Ducourthial, B., Cherfaoui, V., & Bonnet, S. 2014. Design and implementation of the vehicular network testbed using wireless sensors. In International Conference on Ad-Hoc Networks and Wireless (pp. 303-316). Springer Berlin Heidelberg.
26. Thun, G., & Velikov, K. 2014. Action Research and Prototype Testbeds: Prioritizing Collaborative Making in Architectural Research. In ARCC Conference Repository.
27. Kim, O. T. T., Nguyen, V., & Hong, C. S. 2014. Which network simulation tool is better for simulating Vehicular Ad-hoc network?. 2014 Corea Ciencias de la Información Sociedad de la, 41, 930-932.
28. Caban, S. 2012. Evaluation of HSDPA and LTE: from testbed measurements to system level performance. John Wiley & Sons.
29. Tighe, M., Keller, G., Shamy, J., Bauer, M., & Lutfiyya, H. 2013. Towards an improved data centre simulation with DCSim. In Network and Service Management (CNSM), 2013 9th International Conference on (pp. 364-372). IEEE.
30. Zand, P., Mathews, E., Havinga, P., Stojanovski, S., Sisinni, E., & Ferrari, P. 2014. Implementation of wirelesshart in the ns-2 simulator and validation of its correctness. Sensors, 14(5), 8633-8668.
31. Proulx, B., Kuperman, G., Jones, N. M., & Goff, T. 2017. Simulation and modeling of a new medium access control scheme for multi-beam directional networking. In Aerospace Conference, 2017 IEEE (pp. 1-9). IEEE.

32. Ahmed, I., Roussev, V., Johnson, W., Senthivel, S., & Sudhakaran, S. (2016, December). A SCADA System Testbed for Cybersecurity and Forensic Research and Pedagogy. In Proceedings of the 2nd Annual Industrial Control System Security Workshop (pp. 1-9). ACM.
33. Wu, H., & Chen, Z. 2015. SimMon: A Toolkit for Simulating Monitoring Mechanism in Cloud Computing Environments. In Service-Oriented Computing: 13th International Conference, ICSOC 2015, Goa, India, November 16-19, 2015, Proceedings (Vol. 9435, p. 477). Springer.
34. Bonnet, S. 2015. Design and Implementation of the Vehicular Network Testbed Using Wireless Sensors. In Ad-hoc Networks and Wireless: ADHOC-NOW 2014 International Workshops, ETSD, MARSS, MWaoN, SecAN, SSPA, and WiSARN, Benidorm, Spain, June 22--27, 2014, Revised Selected Papers (Vol. 8629, p. 303). Springer.
35. Bonnet, S. 2015. Design and Implementation of the Vehicular Network Testbed Using Wireless Sensors. In Ad-hoc Networks and Wireless: ADHOC-NOW 2014 International Workshops, ETSD, MARSS, MWaoN, SecAN, SSPA, and WiSARN, Benidorm, Spain, June 22--27, 2014, Revised Selected Papers (Vol. 8629, p. 303). Springer.

AUTHORS FROFILE

Venkateswara Raju K, He has an over 14 years' of experience in the field of mobile software development (both Android and IOS). He currently works as a senior Technology Manager in Bangalore, India where the group is working on a new generation e-commerce mobile applications for both IOS and android platforms. He developed several IoT products (thermostat, leak detector & awareness camera) for home/building automation for Honeywell. currently a Ph.D. candidate at REVA University. He is member of IEEE.

Dr Manjula R. Bharamagoudra, Dr. Manjula R. Bharamagoudra has 14+ years of Teaching, Research and Administrative experience. She has many research publications in reputed national/international journals and conferences. Some of the journals where her research articles published are Elsevier, Springer and Wiley, having good impact factors. She has published 12 papers in peer reviewed national and international journals, 18 papers in reputed national and international conferences and 2 book chapter/book. She has published a patent in the year 2018. As per Google Scholar, she has more than 182 citations (h-index = 8 and i-10index = 4) (as on Feb. 2019). She is a reviewer for IEEE conferences and peer reviewed journals such as Elsevier, Wiley, Springer, IEEE systems and so on. She has been awarded with best Innovator - Edupreneur from University Industry Interaction Center REVA University. She is working in the areas related to study of underwater environment, providing solutions to the medical domain, robotics, underwater through internet of things. She is currently guiding three Ph.D. students from REVA University in the areas Internet of Things, Underwater IoT and Antennas. She is a member IETE (IETE) India, member ISTE (MISTE) India and member of IEEE (MIEEE), USA and is actively involved in motivating students and scholars to work on Innovative projects and product development.