Scalable and Interactive Simulation for IoT Applications with TinySim

Gonglong Chen, Wei Dong, Fujian Qiu, Gaoyang Guan, Yi Gao, and Siyu Zeng
College of Computer Science, Zhejiang University.
Alibaba-Zhejiang University Joint Institute of Frontier Technologies.
Email: *{desword, dongw, qiufj, guangy, gaoyi, 21860434}@zju.edu.cn*

***Abstract*—Modern IoT applications are characterized by three important features, i.e., the device heterogeneity, the long-range communication and the cloud-device integration. The above features cause difficulties for IoT application developers in predicting and evaluating the performance of the entire system. To tackle the above difficulties, we design and implement an IoT simulator, TinySim, which satisfies the requirements of high fidelity, high scalability, and high interactivity. Many virtual IoT devices can be simulated by TinySim at the PC end. These IoT devices can send or receive messages from the cloud or smartphones, making it possible for the developers to evaluate the entire system without the actual IoT hardware. We connect TinySim with Unity 3D to provide high interactivity. We design an approximation-based approach to reduce the amount of simulation events, greatly speeding up the simulation process. We conduct extensive experiments to evaluate the performance of TinySim. Results show that TinySim can achieve high accuracy with an error ratio lower than 9.52% in terms of energy and latency. Further, TinySim can simulate 4,000 devices within 11.2 physical-minutes for 10 simulation-minutes, which is about 3× faster than the state-of-art approach.**

I. Introduction

The recent years have witnessed the rapid development of IoT (Internet of Things) technologies and applications. Developers are often confronted with difficulties in implementing a real-world IoT application. For example, developers may have difficulties in evaluating the entire application before the IoT devices are designed and deployed. Unlike PCs and smartphones, IoT devices are special ones which vary with different applications. Hence, the design takes time. The above difficulties would be tackled by using an accurate and scalable IoT simulator. With its availability, IoT developers can quickly simulate the entire applications and evaluate the feasibility of their innovative ideas.

In this poster, we aim to design and implement a simulator for modern IoT applications satisfying the following requirements: **High fidelity**. The simulator should capture the device-level behaviors in a fine-grained manner. Otherwise, it is impossible to evaluate the timing and energy performance of the system. **High scalability.** Future IoT system would consist of a vast number of IoT devices. The simulator should execute at a high speed and can scale to many IoT devices. **High interactivity.** The simulator should provide a user-friendly debug and program UI. Developers can conveniently create scenarios or change environments to trigger events to verify the functionality of IoT applications.

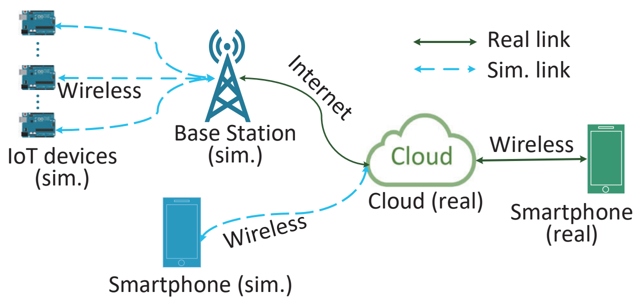


Fig. 1: Network architecture of simulating entire IoT applications in TinySim.

We have designed and implemented TinySim, to meet the above requirements. To use TinySim, the developer writes a device code in a hardware-independent language, TinyLink language, which can be directly used for our simulation. We map TinyLink code into hardware-dependent instructions without actually running them. This approach allows us to obtain *high fidelity* without overhead to really execute the instructions. *To increase scalability*, we propose an approximation based approach to reduce time-consuming events. The behaviours of the time-consuming events are trained using a machine algorithm, e.g., LSTM (Long Short-Term Memory) [1]. The results (e.g., the transmission delay) after executing events are predicted by the machine algorithm. The simulation speed is further sped up by distributing events to many machines. *To provide the high interactivity*, we connect TinySim with a powerful gaming engine Unity 3D. The different simulation speed of TinySim and Unity 3D leads to the large overhead of the event synchronization. To deal with this problem, we propose a dependence graph-based approach to reduce the synchronization overhead while maintaining a good programming experience.

II. TinySim Overview and Main Module

Fig. 1 depicts the network architecture of TinySim, which covers most of a typical IoT application, i.e., the devices, the base station, the cloud and the smartphone. **Between devices and the base station.** Currently, TinySim supports the two most promising LPWAN technologies, i.e., NB-IoT and LoRaWAN. **Between the base station and the cloud.** It is achieved by using the real internet via wired connections between the PC and the cloud. **Between the smartphone and**

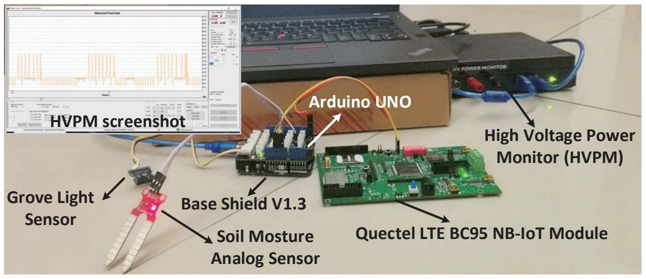

HVPM screenshot
Arduino UNO
High Voltage Power Monitor (HVPM)
Grove Light Sensor
Base Shield V1.3
Soil Mosture Analog Sensor
Quectel LTE BC95 NB-IoT Module

Fig. 2: Monitoring power consumption using HVPM.

TABLE I: Relative error of simulation.

Benchmark	Power (mJ)			Delay (s)		
	Sim.	**Meas.**	**Err**	**Sim.**	**Meas.**	**Err**
Upload	906.43	866.08	4.66%	0.72	0.69	4.53%
Require (SNR=5dB)	194.54	201.67	-3.54%	1.07	1.1	-2.51%
Voice control	19000.00	19983.23	-4.92%	2.00	2.1	-4.76%

the cloud. Developers can control the simulated nodes via an Android application, or generate controlling messages at the PC end.

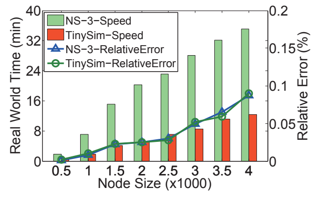
Fidelity. To achieve this, TinySim simulates four main hardware components. *For the mainboard, the sensor and display module*, TinySim captures MCU's timing behaviour by translating the source codes to the MCU instructions. The elapsed time is calculated based on the mapping between MCU instructions and execution time documented in datasheet. The power consumption is then inferred by multiplying the timing behaviours with the corresponding states. *For the communication module.* TinySim carefully simulates the communication behaviours, e.g., the transmission collisions in the random access process in NB-IoT.

Virtual scenario creation. TinySim is connected with Unity 3D, a powerful cross-platform game engine. To reduce the synchronization overhead between TinySim and Unity 3D. We carefully design a dependence graph-based approach. For example, when there is event mismatch between TinySim and Unity 3D, we only need to roll back and re-execute partial events following the dependency graph instead of all events.

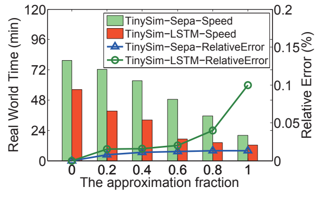
Scalability. One of the main factors that affect the simulation scalability is the number of time-consuming simulation events, which is increased with the number of simulated nodes. TinySim utilizes a machine learning-based approach to reduce the number of simulation events. Specifically, by learning the transmission behaviour of every transmission link, the number of transmissions and the related statistics (e.g., the delay and the power consumption) for one transmission can be directly obtained without actually executing.

III. Evaluation and Conclusion

Hardware. We use two real IoT applications to evaluate the fidelity of TinySim. One is the smart flower spot application. It periodically uploads the sampled data and responses to the required data. Another application is the voice-controlled LED lamp which is the same with [2]. To accurately measure the power consumption, we use HVPM (High Voltage Power Monitor) of Monsoon as shown in Fig. 2. **High Fidelity.** Table I shows that TinySim can accurately simulate the power

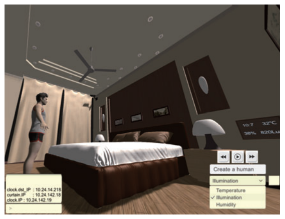


(a) Simulation scale.

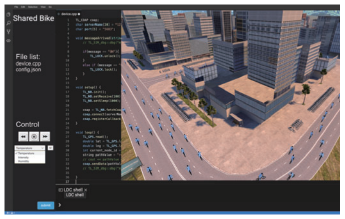


(b) Approximation fraction.

Fig. 3: Scalability (10 simulation minutes, 4000 nodes).



(a) The smart home application.



(b) The shared bike application.

Fig. 4: High interactivity with Unity 3D.

consumption and the action delay with a low average error rate (e.g., 3.8% for the power consumption and 3.95% for the action delay). **High scalability.** Fig. 3(a) shows the results of comparing TinySim with the most related simulator ns-3 [3].Results show that TinySim achieves a faster simulation speed than ns-3 thanks to the simulation approximation and enabled distributed simulation. To evaluate the impact of the approximation fraction on TinySim, we replace different machine learning algorithms of TinySim, e.g., LSTM [1] (TinySim-LSTM), train separate models for different metrics (TinySim-Sepa). Fig. 3(b) shows that TinySim-LSTM can achieve 47.6% faster simulation speed with lower than 5% accuracy degradation compared with TinySim-Sepa. **High interactivity.** Fig. 4(a) presents developers can interactively program and debug a typical smart home application with TinySim. Fig. 4(b) shows a shared bike application and TinySim is integrated with an online IDE.

In this poster, we present TinySim, an IoT simulator for providing entire support to IoT applications. TinySim satisfies the requirements of high fidelity, high scalability, and high interactivity. The future work includes two directions. First, extending TinySim with more communication protocols. Second, extending TinySim with core network simulation.

IV. Acknowledgements

This work is supported by the National Key R&D Program of China under Grant No. 2019YFB1600700, the National Science Foundation of China under Grant No. 61772465, Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020001.

References

[1] H. Sepp and S. JÃijrgen, "Long short-term memory," in *Neural Computation*, vol. 9, no. 8, 1997, pp. 1735–1780.

[2] G. Guan, W. Dong, Y. Gao, K. Fu, and Z. Cheng, "TinyLink: A Holistic System for Rapid Development of IoT Applications," in *Proc. of ACM MobiCom*, 2017.

[3] A. K. Sultania, C. Delgado, and J. Famaey, "Implementation of NB-IoT Power Saving Schemes in Ns-3," in *Proc. of the Workshop on Next-Generation Wireless with Ns-3*, 2019.