Evaluating the Efficiency of Blockchains in IoT with Simulations

Jari Kreku, Visa Vallivaara, Kimmo Halunen and Jani Suomalainen
VTT Techical Recearch Centre of Finland, Finland

Keywords: Internet of Things, Trust, Blockchain, Efficiency, Simulation.

Abstract: As blockchain technology has gained popularity in many different application areas, there is a need to have tools for prototyping and evaluating various ways of applying blockchains. One interesting venue where this type of evaluation is very important is Internet of Things (IoT). In IoT scenarios the efficiency in energy consumption and also the timeliness of the transactions on the blockchain are important variables to consider. We present a way to apply an existing simulation tool - ABSOLUT - in evaluating blockchain implementations on embedded devices. We show the results of simulations on Raspberry Pi and Nvidia Jetson Tk1 platforms and compare the latter to actual executions. Our tool receives a fairly small error (9% on the average) and we see it as a great way to help in deciding the parameters for different blockchain implementations.

1 INTRODUCTION

The invention and success of Bitcoin (Nakamoto, 2008) has brought *blockchain technology* to the forefront of many digitalisation efforts. Blockchains can be seen as distributed, decentralised ledgers, that can keep track of any type of transactions. As the data of every transaction is tracked securely with cryptographic guarantees, an adversary cannot afterwards alter transactions on the blockchain or claim that they did not happen. Also the timeframe of a transaction can be traced in the blockchain. As such, they are a great tool in building trust between distrusting parties communicating over untrusted or unreliable channels.

The Bitcoin blockchain is only a single manifestation of this technology and currently a lot of research is improving both the theoretical foundations and the implementations of blockchains. The technology is young and there are many different approaches that can be utilized. The choices are related to the permissions needed to operate on the blockchain, the contents of the possible transactions and the way the proofs are constructed. For example, one could use *Proof of Work (PoW)* as the Bitcoin uses or more recent ideas such as *Proof of Stake* or *Proof of Space* or a combination of these as the consensus mechanism.

Because blockchains can be used in a number of different ways and there are many parameters in the implementation that affect the security, performance and usability of the system, it is difficult to choose an optimal configuration for one. This is especially true in the Internet of Things (IoT) setting, where a huge

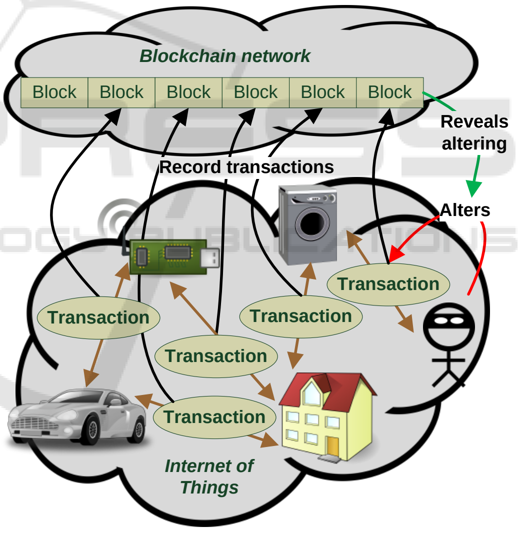

Figure 1: Blockchains for the Internet of Things - Distributed tracking of transactions increases trust by enabling detection of later alterations.

amount of small and resource constrained devices are connected to the Internet. It would be valuable to be able to test and tune the parameters before deploying a blockchain implementation as the update of these systems can be difficult and expensive if not outright impossible. Figure 1 shows a high-level visualisation of a possible blockchain IoT system.

In this paper, we demonstrate how simulation tools can be used to find out the limits and possibilities of blockchain implementations. We adapt VTT's ABSOLUT (Kreku, 2012) simulation tool to test dif-

Kreku, J., Vallivaara, V., Halunen, K. and Suomalainen, J.
Evaluating the Efficiency of Blockchains in IoT with Simulations.
DOI: 10.5220/0006240502160223
In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017)*, pages 216-223
ISBN: 978-989-758-245-5

ferent blockchain alternatives in some use cases. This helps in prototyping blockchains in different environments and finding out which types of instantiations are feasible. It can also be used to find optimal performance (for some parameters) in constrained devices.

The paper is organised as follows: Section 2 describes briefly the background on blockchains and simulation tools. Then we present the methods and settings used. The fourth section contains the results and analysis. Finally, there are discussions, conclusion and problems for future work.

2 BACKGROUND

Our work combines results from two different fields, blockchains and simulation. We briefly present the relevant background work on both of these topics.

2.1 Blockchains

The blockchain technology is the driving force behind Bitcoin – the most successful virtual currency to date. The power of blockchain is in the combination of several ideas that in themselves are not necessary novel, but together provide a great tool for building completely new systems. A blockchain implementation consists of two kinds of records: transactions and blocks. (Franco, Pedro, 2014) A transaction represents an operation or agreement between two parties. Transaction records could e.g. describe money transfers or be used to track products that traverse through their supply chains or the ownership of stock.

The blockchain is a decentralized digital ledger that records transactions between computers so that these cannot be altered retrospectively. The ledger consists of blocks that hold batches of valid transactions. Blocks are added to the ledger by miners. The mining process is made artificially hard in order to enable the blockchain community to securely build a common consensus on accepted transactions. The blockchains can be easily parsed by suitable software to extract relevant information of transactions.

Blockchains can be closed to some permissioned parties or open for everyone to connect to the network, send new transactions to it, verify transactions and create new blocks (permissionless). The design of the Bitcoin blockchain is permissionless and has been the inspiration for other cryptocurrencies and distributed databases. Like many other peer-to-peer applications, these platforms rely on decentralized architectures to build and maintain network applications operated by the community for the community.

The concept of Proof of Work (PoW) first appeared with the aim to combat junk mail (Dwork and Naor, 1993). The idea is to require a user to solve a moderately hard computational puzzle to gain access to the resource. In blockchains, PoW is used to define the average time in which blocks are being generated.

The main idea in Bitcoin's PoW is that each miner calculates a double SHA-256 hash of the block header. The goal here is to find a hash that contains a certain number of leading zeroes. This number is determined by the protocol's difficulty, which is adjusted every 2016 blocks, based on the number of blocks solved and the expected time it should have taken to solve this number of blocks. In Bitcoin, the difficulty is set so that on average every 10 minutes a block is generated by the network (Nakamoto, 2008).

2.1.1 Ethereum

Ethereum's (Wood, 2014) smart contracts are another example of the power of the blockchain. These contracts are programmed to be automatically fulfilled when preconditions are settled. A basic smart contract can hold a contributor's money until a given date or goal is reached. Depending on the outcome, the funds will be released or returned to the contributors.

The Ethereum Frontier network uses a PoW based consensus algorithm Ethash. The reason for constructing a new PoW instead of using an existing one was to tackle the problem of mining centralisation, where a small group miners acquire a large amount of power of the network. (Etherium Project, 2016)

Ethash (Etherium Project, 2016) is based on Dagger Hashimoto algorithm and has an objective of being quickly verifiable with a low memory overhead by light clients, and very quickly mined by using a large amount of memory. Hence, the time of mining is bound to RAM access speed. The main phases of the mining algorithm (Figure 2) are the following:

1. Data sets are generated for verification and for mining. Data sets are derived with hash function (first hashing the block headers to get a seed, then hashing the seed to get a cache and finally hashing the cache to get a big data set). Data sets are generated once at the beginning of mining operations and updated only once in every 30000 blocks. Hence, generating data does not significantly affect the efficiency of the Ethereum blockchain.
2. The main mining function combines and hashes (in Hashimoto-style) a random nonce, a hash of the block header and random slices from the data set. The function loops until a value that satisfies the difficulty threshold is reached (and the PoW is found). For each round the nonce is incremented

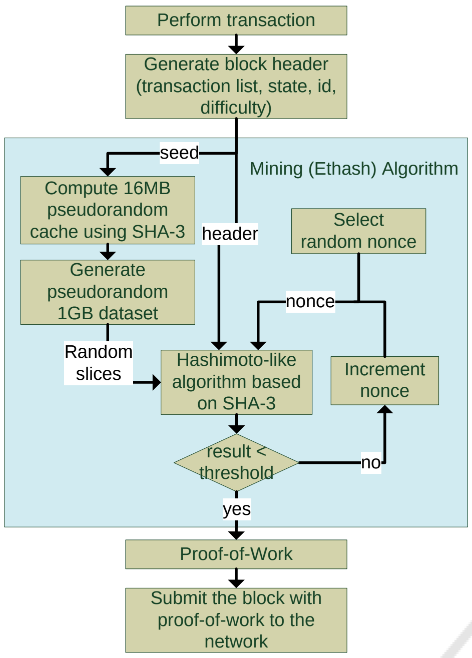
Perform transaction
Generate block header (transaction list, state, id, difficulty)
seed
Mining (Ethash) Algorithm
Compute 16MB pseudorandom cache using SHA-3
header
Select random nonce
Generate pseudorandom 1GB dataset
nonce
Random slices
Hashimoto-like algorithm based on SHA-3
Increment nonce
result < threshold
no
yes
Proof-of-Work
Submit the block with proof-of-work to the network

Figure 2: Phases in the Ethereum mining process.

and new slices are fetched from the memory. The main burden of work in Ethash is in this phase.

3. A block with the PoW is submitted to the network.

The amount of hashing rounds is adjusted dynamically. The number of rounds depends on the difficulty parameter, which in turn depends on the frequency of blocks processed by the Ethereum network. Consequently, the work amount and efficiency of Ethereum mining varies slightly through time.

2.1.2 Efficiency of Blockchains

Optimization of blockchain implementations, esp. cryptocurrencies, has been driven by economic incentives and there are many implementations and optimizations for different platforms. Hashing speed and energy consumption of these have been explored e.g. by the Bitcoin community (Bitcoin community, 2016; Bitcoin community, 2015). Efficiency of blockchains has also been researched by (O'Dwyer and Malone, 2014), who analyzed the economic profitability of Bitcoin mining, and (Ala-Peijari, 2014), who performed a comparative study of performance and energy efficiency of Bitcoin mining on various devices.

The results indicate that Bitcoin-optimised Application Specific Integrated Circuit (ASIC) implementations may achieve over 1000x better efficiency than the mining based on off-the-self desktop CPUs or GPUs. For instance, where regular desktop CPUs and Rasperry Pi were able to achieve efficiency of 0.02 and 0.04 million hashes per joule, respectively, a dedicated ASIC was able to calculate 322.58 million hashes per joule. The efficiency of blockchain implementations was evaluated using hardware test-beds, where blockchains were executed. However, such approaches are inflexible when the software implementation should be evaluated for large amount of different platforms or when new hardware is designed.

A simulation framework for quantifying and comparing the security and performance of different PoW blockchains was introduced by (Gervais et al., 2016). The simulator modeled and evaluated basic blockchain operations and parameters such as block size, block interval and throughput and also enabled modeling of adversarial actions such as double spending and selfish mining. Their analysis covered Bitcon, Litecoin, Dogecoin, as well as Ethereum. The simulator did not model differences and impacts of hardware platforms for the performance of blockchains.

Ethereum's mining algorithm - Ethash - is based on running multiple rounds of Keccak-256 and Keccak-512 hash algorithms, which are (early) variants of SHA-3 standard (NIST, 2015). One of the development criteria for the SHA-3 was the suitability for constrained devices. Hence, the efficiency of the algorithm has been studied on different hardware platforms for IoT, including desktop/laptops running x86, amd64, or ppc32 processors (Bernstein and Lange, 2012), FPGA (Jungk, 2012; Kaps et al., 2012; Latif et al., 2012), ASIC (Guo et al., 2012), RFID (Pessl and Hutter, 2013), and ARM (Schwabe et al., 2012; Bernstein and Lange, 2012). The results highlight that with customised and optimized hardware (FPGAs and ASICs) it is easier to achieve efficiency than with more generic off-the-shelf processors. In addition to hardware platform, the hashing performance and time depends also on the length of the hashed message.

2.2 Simulation

Simulation-based performance and / or power consumption evaluation approaches can be divided into two main categories: full system simulators and single component (e.g. processor, memory) simulators. The full system simulators can be further divided into *virtual system approaches* based on abstract models of both applications and execution platforms, *virtual platform approaches*, which simulate executable applications in functional platform models and *virtual prototype approaches*, which execute real applications in a detailed, low-level platform model.

Both virtual platforms and virtual prototypes are instruction accurate, but virtual platforms are typically coarsely timed and fast to simulate, whereas virtual prototypes are highly accurate but require a lot of modelling effort and are slow to simulate. Virtual systems are not instruction accurate but facilitate low modelling effort and high enough precision for early evaluation and design space exploration.

Gem5 (Binkert et al., 2011) is an instruction accurate simulation system with configurable component models and speed / accuracy tradeoff capability. Nominal simulation speed varies between 300 KIPS and 3 MIPS. SimuBoost (Rittinghaus et al., 2013) is a method for the parallelization of full system simulation based on a virtual platform. Sniper (Heirman et al., 2012) is a parallel multi-core simulator for the performance and power consumption of x86 architectures. It can achieve up to 2 MIPS simulation performance with at most 25% error. VirtualSoC (Bortolotti et al., 2013) is a method for the full system simulation of a general purpose CPU and a many-core HW accelerator using QEMU and SystemC. Virtual system based approaches have been proposed by (Van Stralen and Pimentel, 2010) and (Posadas et al., 2011). COSSIM (Papaefstathiou et al., 2015) is a framework for the full system simulation of networking and processing parts of cyber-physical systems (CPS). It is able to evaluate the performance, power-consumption and security aspects of CPS systems and proposes hardware acceleration with FPGAs to achieve rapid evaluation.

3 MODELLING APPROACH

The ABSOLUT approach (Kreku, 2012) is intended for the evaluation of computer system performance and power consumption in the early phases of design. ABSOLUT uses virtual system modelling, where abstract workload models of applications are simulated on top of performance capacity models of the computing platform (Figure 3). The workload models have basic block, function, process and application layers (Kreku, 2012) and are implemented in SystemC[1]. The basic block layer of the workload models contains abstract instructions read, write and execute and requests for higher-level services such as video decoding or DMA transfer. Tools exist for automatic generation of workload models from traces, measurements, source code or application binaries.

The platform models are also layered and semi-automatically generated from library components and text-based platform description and configuration files. The platform description defines, which components are instantiated from the library and their connections, whereas the configuration file sets up the parameters like clock frequency. The model library includes capacity models of processing unit, interconnect, memory and hardware accelerator components implemented in transaction-level SystemC.

[1]See http://accellera.org for details

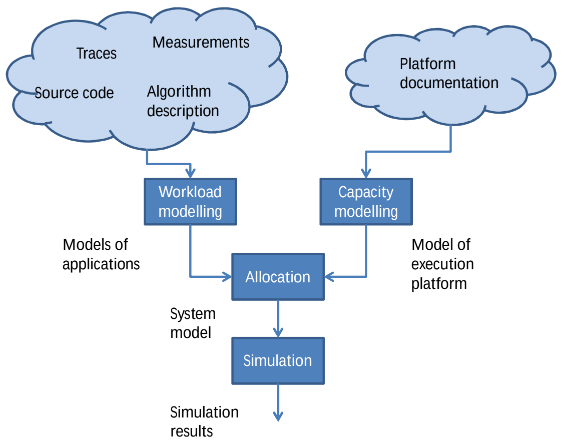

Figure 3: Y-chart diagram of the ABSOLUT performance evaluation approach.

The processing unit models consume the abstract instructions from the workload models. They estimate the time needed to execute the instructions and calculate the resulting utilisation and power consumption. They utilise other components of the system through transactions. High-level services are requested from an OS model and processed by the service provider, which typically is a HW accelerator or a workload model incorporated in the platform.

The modelling approach enables early evaluation, since mature hardware or software is not required for modelling and simulation. ABSOLUT is able to estimate the execution time of an application or a part of an application, the utilisation of the modelled components in the execution platform and the system-level power / energy consumption of the use case.

3.1 Toolset

The ABSOLUT toolset consists of the following: *ABSINTH* is the workload model generator of ABSOLUT. There are multiple versions of *ABSINTH* for modelling from source code (Kreku et al., 2010), application binaries and execution traces. *BEER* is the simulator based on the SystemC simulation kernel. It is also responsible for loading the models and writing the simulation results to data files. *COGNAC* is the platform model generation and configuration tool, which creates the system model from the component models in the model library and the text-based platform description and configuration files. Finally, *VODKA* is used for the visualisation of component

Table 1: Blockchain parameters.

Parameter	Value
Algorithm	PoW (Ethash)
Blockchain software	`geth` version 1.4
Total node amount (n)	1..64
Mining node amount (k)	1..64
Blocks to mine	100..400
difficulty	0x4000
gasLimit	0xffffffff

utilisation and power and energy consumption as a function of time after the simulation has completed.

The average error of the performance estimates produced by ABSOLUT compared to measurements has been 12%. Simulation speed depends on the host processing capacity and the complexity of the system model. Typically, the simulation performance of ABSOLUT has been around 30 to 300 MOPS.

3.2 Target Blockchains and Hardware

We used Ethereum as the blockchain implementation. Table 1 displays the essential parameters of our configuration. The go-language version of Ethereum, `geth`, was used as the application SW. The total number of nodes, n, and the number of mining nodes, k, was varied from 1 to 64. The *difficulty* and *gasLimit* parameters were set up in the `genesis.json` file used to initialise each node. The higher the *difficulty*, the more processing is needed to find a valid new block.

Two alternative execution platforms were selected for evaluation: Raspberry Pi 2 [2] is a cheap general purpose computing platform. It contains 4 low-power ARM Cortex-A7 CPU cores at 900 MHz clock frequency, 1 GB of SDRAM, an ethernet port and other peripherals. Nvidia Jetson TK1[3] development board is based on the Tegra K1 System-on-Chip with 4 high-performance ARM Cortex-A15 cores and 1 low-power ARM core. Tegra's GPU was not used. The effect of network's capabilities were evaluated by using different latency values (from 1 ms to 1000 ms).

4 RESULTS & ANALYSIS

The Ethereum full node – `geth` application – was modelled using the ABSINTH workload model generator. One set of models was generated for each pair of total nodes n and mining nodes k (section 3.2) resulting in a total of 15 sets of workload models.

[2]https://www.raspberrypi.org

[3]http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

The workload models were allocated on the four CPU cores in the ABSOLUT capacity models of the Raspberry Pi 2 and Jetson TK1 execution platforms. The same workload models were used on both platforms. The resulting system models were simulated and the execution time, average power and total energy needed to mine 400 blocks was extracted.

4.1 Performance and Power

The results from the performance and power consumption simulations are listed in Table 2. These simulations assume that the network latency between the nodes is minimal (<1 ms). Both the total number of nodes and the number of mining nodes varied from 1 to 16 on both platforms. The results indicate that while the execution time to mine 400 blocks decreases as the number of mining nodes is increased, the total energy consumption is the smallest with only 1 node. Thus, it is best to use the least number of nodes with which the performance requirements are satisfied.

Comparison of the platforms reveals that e.g. one Jetson TK1 is able to achieve a bit better performance than four Raspberry Pi 2 nodes – with a higher power but smaller energy consumption. The Pi 2 platform is a much better choice for the non-mining nodes as their utilisation is low and average power is close to idle power. Even the Pi 2 has plenty of processing capacity left to collect sensor data, for example.

When the number of mining nodes is increased, the average power of the mining nodes decreases as the parallel part of the program execution becomes shorter. The inverse happens to the non-mining nodes due to messages of new blocks appearing more often.

4.2 Effect of Network Latency

The results in section 4.1 show that the increase in mining performance is almost linear as the number of nodes is increased – at least when at most 16 nodes is used. However, those simulations assumed that the network latency between the nodes is minimal. When the latency is increased, it will take more time before the mining nodes become aware of new blocks and PoWs produced by the other miners.

To analyse the effect of network latency, another set of simulations was performed. Table 3 depicts, how the increasing network latency affects the execution time, power and energy required to mine the 400 blocks with a given number of nodes.

The simulations show that the network latency does not affect the mining performance considerably as long as the it stays below 100 ms. After 300-400

Table 2: Use cases and their estimated execution time for mining 400 blocks.

Platform	Nodes		Execution time	Average power		Energy
	Total (n)	Mining (k)		k nodes	(n-k) nodes	
Raspberry Pi 2	1	1	3082 s	1724 mW	0 mW	5.3 kJ
	2	1	3082 s	1724 mW	1144 mW	8.8 kJ
	2	2	1710 s	1704 mW	0 mW	5.8 kJ
	4	1	3082 s	1724 mW	1143 mW	15.9 kJ
	4	2	1710 s	1704 mW	1144 mW	9.7 kJ
	4	4	954 s	1667 mW	0 mW	7.5 kJ
	8	1	3082 s	1724 mW	1143 mW	30.0 kJ
	8	2	1710 s	1704 mW	1144 mW	17.6 kJ
	8	4	954 s	1667 mW	1145 mW	10.7 kJ
	8	8	575 s	1612 mW	0 mW	7.4 kJ
	16	1	3082 s	1724 mW	1143 mW	58.1 kJ
	16	2	1710 s	1704 mW	1144 mW	33.2 kJ
	16	4	954 s	1667 mW	1145 mW	19.5 kJ
	16	8	575 s	1612 mW	1146 mW	12.7 kJ
	16	16	386 s	1544 mW	0 mW	10.0 kJ
Nvidia Jetson TK1	1	1	702 s	6517 mW	0 mW	4.6 kJ
	2	1	702 s	6517 mW	1990 mW	6.0 kJ
	2	2	393 s	6308 mW	0 mW	5.0 kJ
	4	1	702 s	6517 mW	1973 mW	8.7 kJ
	4	2	393 s	6308 mW	2006 mW	6.5 kJ
	4	4	223 s	5948 mW	0 mW	5.3 kJ
	8	1	702 s	6517 mW	1968 mW	14.2 kJ
	8	2	589 s	6308 mW	1990 mW	9.7 kJ
	8	4	223 s	5948 mW	2032 mW	7.1 kJ
	8	8	138 s	5435 mW	2184 mW	6.3 kJ
	16	1	702 s	6517 mW	1966 mW	25.3 kJ
	16	2	393 s	6308 mW	1985 mW	15.9 kJ
	16	4	223 s	5948 mW	2018 mW	10.7 kJ
	16	8	138 s	5435 mW	2071 mW	8.3 kJ
	16	16	96 s	4838 mW	0 mW	7.6 kJ

ms the performance starts to drop, and e.g. with 4 mining nodes and a latency of 1000 ms the execution time has increased by 80%. Energy consumption follows the increase in execution time since average power consumption is practically the same regardless of the network latency.

4.3 Validation of Simulation Results

To evaluate the precision of the simulations a subset of the Jetson TK1 use cases was both simulated and then measured in real hardware. The execution time to mine a specific number of blocks was extracted from the simulations with a varying number of mining nodes and network latency. The results were compared to the values obtained from the development board. Due to a bug[4] in the `geth` Ethereum client we were unable to validate the simulation results on Raspberry Pi 2. Measurements of multiple nodes were performed by running multiple instances of `geth` on a single Jetson TK1. Therefore, only one thread per node was used in most comparisons to ensure that each node had its own core. Network latency was emulated with linux traffic control (`tc`).

Table 4 compares the execution times estimates from simulations to observations from the real execution platform. The average and maximum absolute errors in simulations were 9% and 14%, respectively. The average error is slightly better than the historical average error of 12% in ABSOLUT simulations. Individual results indicate that the effect of network latency and multithreading in `geth` is underestimated.

[4]https://github.com/ethereum/go-ethereum/issues/2783

Table 3: Execution time and energy consumption as a function of network latency and number of mining nodes.

Mining nodes	Latency	Execution time	Energy
4	1 ms	956 s	6.4 kJ
4	10 ms	967 s	6.4 kJ
4	100 ms	1030 s	6.9 kJ
4	1000 ms	1710 s	11.7 kJ
16	1 ms	386 s	9.5 kJ
16	10 ms	390 s	9.6 kJ
16	100 ms	406 s	10.0 kJ
16	1000 ms	576 s	14.8 kJ
64	1 ms	245 s	22.3 kJ
64	10 ms	245 s	22.4 kJ
64	100 ms	249 s	22.8 kJ
64	1000 ms	292 s	27.6 kJ
1	1 ms	3082 s	5.3 kJ

Table 4: Simulation vs. measurements on the Tegra K1 platform. N, T, and B are the number of nodes, threads and blocks. L is the network latency and E the simulation error.

N	T	B	L [ms]	Exec. time [s] Est.	Exec. time [s] Meas.	E [%]
1	1	100	1	621	642	-3
1	1	200	1	1189	1122	6
1	1	400	1	2325	2314	0
2	1	200	1	650	569	14
4	1	200	1	352	384	-9
4	1	200	10	357	415	-14
4	1	200	100	383	443	-14
4	1	200	1000	658	736	-11
1	2	200	1	660	580	14
1	3	200	1	472	451	5
1	4	200	1	378	373	1
Average absolute error						9%
Maximum absolute error						14%

5 DISCUSSION

In the different kind of blockchain applications the parameters can differ greatly. The average confirmation time of a Bitcoin transaction is over 10 minutes and an hour in the worst case, which would not work in many practical implementations. Also if the difficulty of the PoW takes too much computing power it cannot to be used with smaller gadgets. Possibility to simulate different situations with chosen parameters saves a lot of developers time and resources.

Mining algorithms, including Ethereum's SHA-3 based Ethash, are optimal for dedicated hardware platforms. However, for IoT such requirement is not feasible; additional dedicated hardware would increase the cost of devices significantly. Hence, blockchains must operate on general purpose platforms, like ARM in our simulations. However, it is often possible to assign mining responsibilities for high-end machines and deploy only light transaction verification functionality to the constrained things. The simulation tool presented in this paper provides a mean to asses, if it is possible to deploy blockchains that rely only on the constrained things for mining.

Blockchains where mining relies on constrained things may be vulnerable for attack where an adversary with high mining capabilities can surpass the performance of very large number of things. Thus the consensus mechanism should be designed in a way that prevents such attackers from benefiting from their superior power. Also permissions can be used to limit the possibility of an attacker on the blockchain.

6 CONCLUSION

Our work demonstrates that we can use the ABSOLUT tool to quite accurately simulate blockchain implementations in embedded devices. This should help in rapid prototyping of different blockchains on IoT platforms and bring savings in projects that need to develop blockchain implementations.

As future work, we seek to utilise this tool in our current and upcoming projects related to blockchains and IoT. This will help in refining the tools and also further validating our approach. We also expect this tool to help in getting better results in these projects.

ACKNOWLEDGEMENTS

The authors wish to thank TEKES - The Finnish Funding Agency for Technology and Innovation (BOND project) for supporting this research.

REFERENCES

Ala-Peijari, O. (2014). Bitcoin The Virtual Currency: Energy Efficient Mining of Bitcoins. Master's Thesis, Aalto University.

Bernstein, D. J. and Lange, T. (2012). The new SHA-3 software shootout. *IACR Cryptology ePrint Archive*, 2012:4.

Binkert, N., Beckmann, B., and Black, G. (2011). The Gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39:1–7.

Bitcoin community (2015). Bitcoin wiki: Non-specialized hardware comparison. https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison. Accessed 2016-11-14.

Bitcoin community (2016). Bitcoin wiki: Mining hardware comparison. https://en.bitcoin.it/wiki/Mining_hardware_comparison. Accessed 2016-11-14.

Bortolotti, D., Pinto, C., and Marongiu, A. (2013). VirtualSoC: A full system simulation environment for massively parallel heterogeneous System-on-Chip. In *Parallel and Distributed Processing Symposium Workshop & PhD Forum*.

Dwork, C. and Naor, M. (1993). Pricing via Processing, Or, Combatting Junk Mail, Advances in Cryptology. In *CRYPTO'92: Lecture Notes in Computer Science No. 740*, page 139–147. Springer.

Etherium Project (2016). Ethash specification. https://github.com/ethereum/wiki/wiki/Ethash. Accessed 2016-11-14.

Franco, Pedro (2014). *Understanding Bitcoin: Cryptography, Engineering and Economics.* John Wiley & Sons.

Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., and Capkun, S. (2016). On the security and performance of proof of work blockchains. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*.

Guo, X., Srivastav, M., Huang, S., Ganta, D., Henry, M. B., Nazhandali, L., and Schaumont, P. (2012). ASIC implementations of five SHA-3 finalists. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1006–1011.

Heirman, W., Sarkar, S., and Carlson, T. (2012). Power-aware multi-core simulation for early design stage hardware/software co-optimization. In *Proceedings of the 21st international conference on parallel architectures and compilation techniques*.

Jungk, B. (2012). Evaluation of compact FPGA implementations for all SHA-3 finalists. In *The Third SHA-3 Candidate Conference*.

Kaps, J.-P., Yalla, P., Surapathi, K. K., Habib, B., Vadlamudi, S., and Gurung, S. (2012). Lightweight implementations of SHA-3 finalists on FPGAs. In *The Third SHA-3 Candidate Conference*. Citeseer.

Kreku, J. (2012). Early-phase Performance Evaluation of Computer Systems using Workload Models and SystemC.

Kreku, J., Tiensyrjä, K., and Vanmeerbeeck, G. (2010). Automatic Workload Generation for System-level Exploration based on Modified GCC Compiler. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*.

Latif, K., Rao, M. M., Aziz, A., and Mahboob, A. (2012). Efficient hardware implementations and hardware performance evaluation of SHA-3 finalists. In *The Third SHA-3 Candidate Conference*.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcointalk.org/bitcoin.pdf.

NIST (2015). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. FIPS PUB 202. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

O'Dwyer, K. and Malone, D. (2014). Bitcoin mining and its energy footprint. In *IET Irish Signals & Systems Conference*.

Papaefstathiou, I., Chrysos, G., and Sarakis, L. (2015). COSSIM: A Novel, Comprehensible, Ultra-Fast, Security-Aware CPS Simulator.

Pessl, P. and Hutter, M. (2013). Pushing the limits of SHA-3 hardware implementations to fit on RFID. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 126–141. Springer.

Posadas, H., Real, S., and Villar, E. (2011). M3-Scope: Performance modelling of multi-processor embedded systems for fast design space exploration.

Rittinghaus, M., Miller, K., Hillenbrand, M., and Bellosa, F. (2013). SimuBoost: Scalable Parallelization of Functional System Simulation. In *11th International Workshop on Dynamic Analysis*.

Schwabe, P., Yang, B.-Y., and Yang, S.-Y. (2012). SHA-3 on ARM11 processors. In *International Conference on Cryptology in Africa*, pages 324–341. Springer.

Van Stralen, P. and Pimentel, A. (2010). Scenario-based design space exploration of MPSoCs. In *Proceedings of the IEEE International Conference on Computer Design*.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*.