Evaluating the Efficiency of Blockchains in IoT with Simulations

Conference Paper · April 2017

DOI: 10.5220/0006240502160223

Kimmo Halunen
VTT Technical Research Centre of Finland
36 PUBLICATIONS **80** CITATIONS
SEE PROFILE

Visa Vallivaara
VTT Technical Research Centre of Finland
14 PUBLICATIONS **20** CITATIONS
SEE PROFILE

Jani Suomalainen
VTT Technical Research Centre of Finland
41 PUBLICATIONS **231** CITATIONS
SEE PROFILE

Evaluating the Efficiency of Blockchains in IoT with Simulations

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 (7% 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 an transaction can be traced in the blockchain. As such, they are a great tool in building trust and trustworthy systems between mutually distrusting parties communicating over untrusted and even unreliable channels.

The original Bitcoin blockchain is only a single manifestation of this technology and currently a lot of research is improving both the theoretical foundations and the possible implementations of blockchains. Because the technology is still very young, there are many different approaches that can be utilized. These 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 the *Proof of Work (PoW)* that the Bitcoin uses (or some other work function) or more recent ideas such as *Proof of Stake* or *Proof of Space* or some 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 many times very difficult to choose an optimal configuration for one's blockchain. This is especially true in the Internet of Things (IoT) setting, where a huge amount of small and resource constrained devices are connected to the Internet. In this type of scenario, 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.

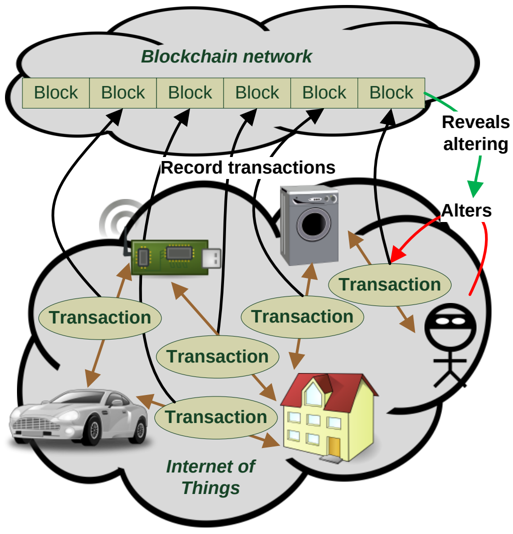

Figure 1: Blockchains for the Internet of Things - Distributed tracking of transactions increases trust by enabling detection of later alterations.

In this paper, we demonstrate how simulation tools can be used to find out the limits and possibilities of blockchain implementations. We use and adapt an existing simulation tool - ABSOLUT (Kreku, 2012) - to test different blockchain alternatives in some use cases. This should help in prototyping blockchains in many different environments and to find 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 in the following way. In the next section we describe briefly the background on blockchains and simulation tools. Then we present the methods and tools that we used in our simulations and we also describe the settings that we used to test the simulation. The fourth section contains the results and their analysis. We finish with some discussion on our findings and by giving our conclusion and problems for future work.

2 Background

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

2.1 Blockchains

As mentioned in the introduction, the blockchain technology is the driving force behind Bitcoin, which is 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 in combination provide a great tool for building completely novel systems. A blockchain implementation consists of two kinds of records: transactions and blocks. (Franco, Pedro, 2014) A transaction record represents an operation or agreement between two parties. Transaction records could, for instance, describe money transfers, but they may also describe movements of things or assets e.g. transactions can be used to track products that traverse through their supply chains from factories to end-users or as a way to track ownership of stock in a stock exchange.

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 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.

Cryptocurrencies such as Bitcoin have been the first to widely adopt blockchain technology. These tend to provide similar service as any currency, but with varying levels of success. A great survey on the developments of cryptocurrencies, which have been the first and foremost application domain of blockchains can be found in (Bonneau et al., 2015). Many of the topics covered in (Bonneau et al., 2015) are also of relevance to the wider use of blockchains such as smart contracts.

In general, blockchains can be closed to only some permissioned parties or permissionless, where they are open for everyone to connect to the network, send new transactions to it, verify transactions and create new blocks. The latter is also the original design of the Bitcoin blockchain and has been the inspiration for other cryptocurrencies and distributed databases. Like many other peer-to-peer applications, these platforms all rely on decentralized architectures to build and maintain network applications that are operated by the community for the community.

One thing of note is the claim of anonymity in the Bitcoin system. At first Bitcoin was touted as an *anonymous* system of transfering funds. However, this has not been true even though it can be difficult to trace a bitcoin identity (a public key) to some real world identity. The blockchain actually keeps a record of all the transactions of a given identity and this can be used with external information about the transactions to solve the real identity. Some alternatives such as zcash[1] and Darkcoin (Duffield and Hagan, 2014) have emerged that promise better anonymity and privacy features.

The concept of Proof of Work (PoW) first appeared in a work by Dwork and Naor (Dwork and Naor, 1993) with the aim to combat junk mail. The idea is to require a user to solve a moderately hard computational puzzle in order to gain access to the resource. The term PoW was coined in 1999 by Jakobsson and Juels (Jakobsson, 1999), in which the notion of PoW is being formalized. PoW is used in many blockchain based cryptocurrencies 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 has a particular characteristic, namely, it contains a certain number of leading zero's. 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. PoW therefore provides a timestamping mechanism (Nakamoto, 2008) for the content of a block, such as transactions.

[1] see `https://z.cash` for details

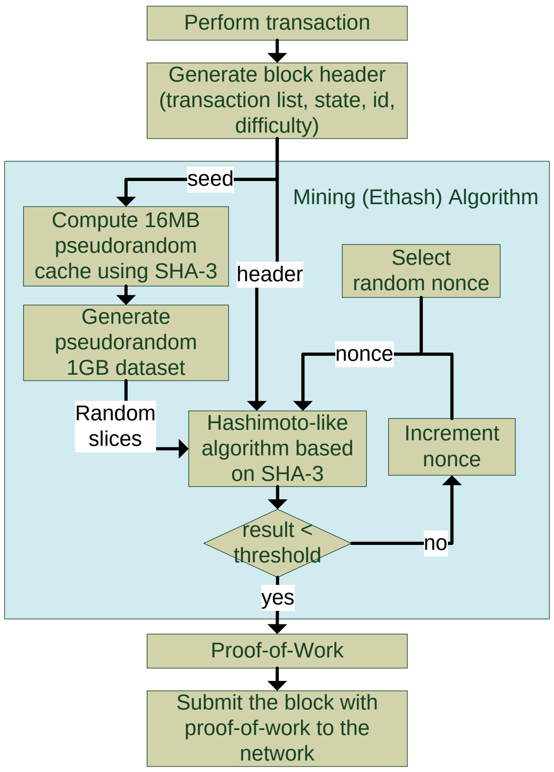
Perform transaction
Generate block header (transaction list, state, id, difficulty)
seed
Mining (Ethash) Algorithm
Compute 16MB pseudorandom cache using SHA-3
header
Select random nonce
Generate pseudorandom 1GB dataset
nonce
Random slices
Hashimoto-like algorithm based on SHA-3
Increment nonce
result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

Figure 2: Phases in the Ethereum mining process

2.2 Ethereum

Smart contracts such as provided by Ethereum (Wood, 2014) are another example of the power of the blockchain. These contracts are basically programmed to be automatically fulfilled when some preconditions are settled. A basic smart contract can hold a contributor's money until any given date or goal is reached. Depending on the outcome, the funds will either be released or returned back to the contributors. All of this is possible without requiring a centralized arbitrator, clearing house or having to trust anyone.

The Ethereum Frontier network also currently uses a Proof of Work based consensus algorithm. The PoW algorithm used in the Frontier is called Ethash and it was designed specifically for Ethereum. The main reason for constructing a new PoW instead of using an existing one was to tackle the problem of mining centralisation, where a small group mining operators acquire a disproportionately 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 - illustrated in Figure 2 - are the following:

1. Data sets are generated for verification (16MB cache) and for mining (1 GB). 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 only once at the beginning of mining operations and updated only once in every 30000 blocks. Hence, generating data does not affect significantly to 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 and new slices are fetched from the memory. The main burden of work in Ethash is in this phase.
3. A block with the proof-of-work is submitted to the Ethereum 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 within time.

2.3 Efficiency of blockchain implementations

Optimization of blockchain implementations, particularly cryptocurrencies, has been driven by economic incentives and there exists a large amount of implementations and optimizations for different platforms. Hashing speed and energy consumption of these implementations has been explored e.g. by the Bitcoin community (Bitcoin community, 2016; Bitcoin community, 2015). Efficiency of blockchains has also been researched by O'Dwyer et al. (O'Dwyer and Malone, 2014), who analyzed economic profitability of Bitcoin mining, and Ala-Peijari (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 thousand times better efficiency than the mining based on off-the-self desktop CPUs or graphics card GPUs. For instance, where regular desktop CPUs and ARM 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, in these approaches, evaluated using real hardware testbeds 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.

Etherium'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-self processors. In addition to hardware platform, the hashing performance and time depends also on the length of the hashed message.

2.4 Simulation of performance and power consumption

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 very coarsely timed and fast to simulate, whereas virtual prototypes are highly accurate with respect to timing 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 a modular, instruction accurate simulation system with configurable CPU, cache and interconnect models and speed / accuracy tradeoff capability. Nominal simulation speed of Gem5 varies from around 300 KIPS to 3 MIPS. SimuBoost (Rittinghaus et al., 2013) is a method for the parallelization of functional full system simulation based on a virtual platform like QEMU[2]. Sniper (Heirman et al., 2012) is a parallel multi-core simulator for the performance and power consumption of x86 architectures. It is capable of achieving up to 2 MIPS simulation performance with at most 25% error in results. 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 based on abstract models of applications to facilitate rapid early evaluation 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. The work by (Hockner et al., 2013) presents a static design space exploration method for the analysis of timing and power consumption of CPS systems.

3 Modelling approach

In this section we describe the methods that we utilised in our simulations. In the core of our system is the ABSOLUT modelling and simulation approach (Kreku, 2012) for the evaluation of computer system performance and power consumption in the early phases of design.

[2]See `http://qemu.org` for details

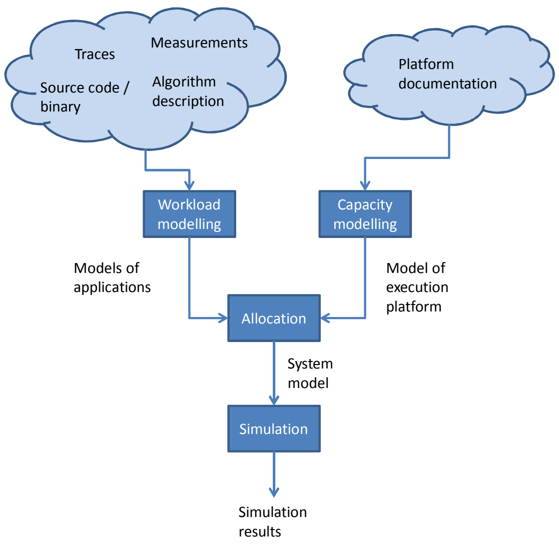

Figure 3: Y-chart diagram of the ABSOLUT performance evaluation approach.

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[3]. However, tools exist for generating workload models from traces, measurement data, source code or application binaries so that the user does not need to write SystemC code manually. The basic block layer of the workload models can contain abstract instructions read, write and execute and requests for higher-level services such as video decoding or DMA transfer.

The platform models are also layered, hierarchical models, which are semi-automatically generated from components in a model library and text-based platform description and configuration files. The platform description file describes, which components are instantiated from the library and how the components are connected, whereas the configuration file sets up the parameter values like clock frequency for the components. The model library includes capacity models of processing unit (e.g. ARM), interconnect (bus, crossbar), memory and hardware accelerator components, which have been implemented in transaction-level SystemC.

In ABSOLUT, the processing unit models consume the abstract instructions from the basic block layer of the workload models. They estimate the time needed to execute those instructions and calculate the resulting utilisation and power consumption. Depending on the instruction, they may involve other components of the system through transactions. High-level services are requested from an operating system model and processed by the provider of the service, which typically is either a hardware accelerator or a service workload model incorporated in the platform.

The modelling approach enables early evaluation, since mature hardware or software is not required for the modelling and simulation of a complete system. 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) (both GCC and LLVM based), application binaries and execution traces.

- *BEER* is the simulator based on the SystemC simulation kernel. Besides the simulation itself, it is responsible for loading the models and writing the simulation results to the screen and output data files.

- *COGNAC* is the platform model generation and configuration tool, which creates the SystemC model of the entire system from the component models in the model library and the text-based platform description and configuration files. *ARMAGNAC* is a subtool for calculating addresses for passing transactions between the components during simulation.

- Finally, *VODKA* is used for the visualisation of component utilisation and power and energy consumption as a function of time after the simulation has completed.

The average difference between the performance estimates produced by ABSOLUT and measurements made in corresponding real systems has been 12% (Kreku, 2012). Simulation speed depends greatly on the processing capacity of the host computer and the complexity of the system model. However, the simulation performance of ABSOLUT has typically been around 30 to 300 Million Operations Per Second (MOPS).

[3]See `http://accellera.org` for details

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

Table 2: Platform parameters

Parameter	Value
Hardware	Raspberry Pi 2, Nvidia Jetson TK1
Network latency	1..1000ms

3.2 Target Blockchains and Hardware

As the blockchain implementation we used the Ethereum blockchain. Table 1 displays the essential parameters of our blockchain 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 (Table 2). The first one is the Raspberry Pi 2 [4], which is a cheap general purpose computing platform. It contains four low-power ARM Cortex-A7 CPU cores at 900 MHz clock frequency, 1 GB of SDRAM memory, an ethernet port and other peripherals. The second platform is the Nvidia Jetson TK1[5] development board based on the Tegra K1 System-on-Chip with 4 high-performance ARM Cortex-A15 cores and 1 low-power ARM core. The Tegra platform contains also a powerful GPU, which could be used to accelerate mining. However, for this case study the GPU was not used and the mining was performed only on the four CPU cores.

4 Results & Analysis

The Ethereum full node – `geth` application – was modelled using the ABSINTH workload model generator (Kreku et al., 2010). One sets of models was generated for each pair of total nodes n and mining nodes k (section 3.2) resulting to a total of 15 sets of workload models.

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 consumption and total energy consumption needed to mine 400 blocks was extracted.

4.1 Performance and power consumption

The results from the performance and power consumption simulations are listed in Table 3. These simulations assume that the network latency between the nodes is minimal, i.e. 1 ms or less. Both the total number of nodes n and the number of mining nodes k varied from 1 to 16 on both platforms. The results indicate that while the execution time to mine 400 blocks decreases each time the number of mining nodes is increased, the total energy consumption is the smallest with only 1 node. Thus, as long as the performance is adequate, it is best to use the least number of nodes.

Comparison of the platforms reveals that e.g. one Jetson TK1 is able to achieve similar performance to four Raspberry Pi 2 nodes – but with smaller power and energy consumption. The Raspberry Pi 2 platform is a much better choice for the non-mining nodes as the utilisation of those nodes is low and average power consumption is close to idle power. Thus, 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 consumption 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 shown in section 4.1 highlight 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. If the network latency is increased, it will take more time before the mining nodes become aware of new blocks and proofs-of-work produced by the other miners.

[4]`https://www.raspberrypi.org`

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

Table 3: 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	3359 s	1735 mW	0 mW	5.8 kJ
	2	1	3359 s	1735 mW	1144 mW	9.7 kJ
	2	2	1858 s	1715 mW	0 mW	6.4 kJ
	4	1	3359 s	1735 mW	1143 mW	17.4 kJ
	4	2	1858 s	1715 mW	1144 mW	10.6 kJ
	4	4	1031 s	1681 mW	0 mW	8.1 kJ
	8	1	3359 s	1735 mW	1143 mW	32.7 kJ
	8	2	1858 s	1715 mW	1144 mW	19.1 kJ
	8	4	1031 s	1681 mW	1144 mW	11.7 kJ
	8	8	617 s	1628 mW	0 mW	8.8 kJ
	16	1	3359 s	1735 mW	1143 mW	63.4 kJ
	16	2	1858 s	1715 mW	1144 mW	36.1 kJ
	16	4	1031 s	1681 mW	1144 mW	21.1 kJ
	16	8	617 s	1628 mW	1146 mW	13.7 kJ
	16	16	411 s	1562 mW	0 mW	10.3 kJ
Nvidia Jetson TK1	1	1	1068 s	6587 mW	0 mW	7.0 kJ
	2	1	1068 s	6587 mW	1776 mW	8.9 kJ
	2	2	589 s	6440 mW	0 mW	7.6 kJ
	4	1	1068 s	6587 mW	1764 mW	12.7 kJ
	4	2	589 s	6440 mW	1788 mW	9.7 kJ
	4	4	326 s	6173 mW	0 mW	8.0 kJ
	8	1	1068 s	6587 mW	1761 mW	20.2 kJ
	8	2	589 s	6440 mW	1776 mW	13.9 kJ
	8	4	326 s	6173 mW	1809 mW	10.4 kJ
	8	8	194 s	5769 mW	0 mW	8.9 kJ
	16	1	1068 s	6587 mW	1759 mW	35.2 kJ
	16	2	589 s	6440 mW	1773 mW	22.2 kJ
	16	4	326 s	6173 mW	1798 mW	15.1 kJ
	16	8	194 s	5769 mW	1841 mW	11.8 kJ
	16	16	128 s	5253 mW	0 mW	10.7 kJ

To analyse the effect of network latency, another set of simulations was performed. Table 4 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 revealed that the network latency does not affect the mining performance considerably as long as the latency stays below 100 ms. After 300-400 ms the performance starts to drop, and e.g. with 4 mining nodes and a network 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 `geth` simulations a subset of the Jetson TK1 use cases was both simulated with ABSOLUT and then measured in the real hardware platform. 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 then compared to the values obtained from the development board. Due to a bug[6] in the `geth` Ethereum client we were unable to validate the simulation results on Raspberry Pi 2 hardware.

Table 5 displays the execution times obtained from simulations and the execution times observed on the real execution platform. The average absolute error in simulations was 7% and the maximum was 13%. The average error is slightly better than the historical average error of 12% in ABSOLUT simulations. Individual results indicate that the single

[6]See `https://github.com/ethereum/go-ethereum/issues/2783` for details of a similar issue

Table 4: 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	1031 s	6.9 kJ
4	10 ms	1045 s	7.0 kJ
4	100 ms	1115 s	7.5 kJ
4	1000 ms	1858 s	12.8 kJ
16	1 ms	411 s	10.2 kJ
16	10 ms	414 s	10.3 kJ
16	100 ms	432 s	10.9 kJ
16	1000 ms	617 s	16.1 kJ
64	1 ms	255 s	23.6 kJ
64	10 ms	256 s	23.7 kJ
64	100 ms	261 s	24.2 kJ
64	1000 ms	307 s	29.4 kJ
1	1 ms	3359 s	5.8 kJ

node performance and the effect of network latency is somewhat underestimated. On the other hand, the performance of multiple nodes in co-operation is overestimated.

5 Discussion

In the different kind of blockchain applications the parameters can differ greatly. The average confirmation time of the Bitcoin transaction is over 10 minutes and in the worst cases can take up to an hour, which would not work in many practical implementations. Also if the difficulty of the proof work 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 (FPGA, ASIC) 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, in many cases, it is 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 whether it is possible to deploy blockchains that rely only on the constrained things for the mining.

Blockchains where mining relies on constrained things may be vulnerable for attack where a single adversary with high mining capabilities can surpass the performance of very large number of things. Thus the PoW (or some other consensu 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 (outside) attacker on the blockchain.

6 Conclusion

Our work demonstrates that we can use the existing 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.

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 compari-

Table 5: Comparison of geth simulation results to measurements on the Nvidia Tegra K1 platform

Nodes	Threads	Blocks	Network latency	Execution time		Error
				Estimated	Measured	
1	1	100	1 ms	716.5 s	671.5 s	7%
1	1	200	1 ms	1371.1 s	1291.7 s	6%
1	1	400	1 ms	2680.3 s	2481.4 s	8%
2	1	200	1 ms	749.2 s	761.8 s	-2%
4	1	200	1 ms	405.6 s	427.2 s	-5%
4	1	200	10 ms	411.6 s	446.7 s	-8%
4	1	200	100 ms	441.5 s	504.7 s	-13%
4	1	200	1000 ms	759.2 s	818.8 s	-7%
Average absolute error						7%
Maximum absolute error						13%

son. 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.

Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. (2015). Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE.

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*.

Duffield, E. and Hagan, K. (2014). Darkcoin: Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system.

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*.

Hockner, B., Hofstedt, P., and Kaltschmidt, S. (2013). Design space exploration for cyber physical system design using constraint solving. In *Forum on Specification & Design Languages (FDL).*

Jakobsson, Markus; Juels, A. (1999). Proofs of Work and Bread Pudding Protocols. In *Communications and Multimedia Security*, page 258–272. Kluwer Academic Publishers.

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 multiprocessor 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*.