Blockchain as a P2P protocol for the Internet of Things

[redacted]

1 Introduction

The initial goal for the project was to introduce a decentralized home system solution in which no central hub is required. Blockchain was a great tool for this type of P2P network since it introduced trustless authentications. In this way the home system need not to worry about external malicious attacks. However, while the blockchain offers a secured and trustless decentralized IoT system, it requires an efficient P2P protocol to accumulate the amount of broadcast messages being created. This issue remains true for any other decentralized networks. As decentralization is the future of IoT, given the huge numbers of devices, we need a P2P protocol that make a good trade-off between security and performance. Interestingly, the blockchain has been adopted to design such protocol. Our project is simply an experiment to test the performance and feasibility of using blockchain for P2P messaging. Our targeted protocol is Whisper, which utilizes the Ethereum blockchain.

2 System Design

We noticed that there are other blockchain networks that support message passing as transactions such as Ethereum. By using these established networks we will be able to demonstrate our idea more easily. Ethereum offered a Java implementation of the software called Ethereumj [1], which enable us to build test application on Android. Instead of using Raspberry Pi microcomputers, we decided to develop the demonstration on Android phones in order to utilize the integrated sensors. Ethereum supports a protocol called Whisper that allows clients to pass messages that does not need to be permanently stored in the blockchain. Our experiment is designed as below:

- One phone, operated as sender, continuously collect data from the light sensor and aggregate data every one second into a "batch". Data will be formatted in json.
- Once every 10 seconds, the sender format the batch into a message compatible with the Whisper API. The sending time is also attached to the message.
- Using the Whisper API, the message is sent over the Ethereum network to the other phone, the receiver.
- The receiver analyzes the data to make sure that the measurements satisfy some conditions and if so, invoke the camera to take a picture. The time the message is received will be recorded.
- The receiver analyzes the data to make sure that the measurements satisfy some conditions and if so, invoke the camera to take a picture. The receiving time is also recorded. We will calculate the elapsed time to evaluate the feasibility of using blockchain as a P2P protocol.

The Whisper protocol plays a key role in maintaining the security and efficiency in sending, multi-casting, and broadcasting messages. The process of inserting (authoring) a messages is as following (details of this process can be found in the reference link):

- Compose data through concatenating the relevant flag byte, a signature of the payload if the user specified a valid author identity, and the user-given payload.
- Encrypt the data if an access ("destination") identity's public key is given by the user.
- Compose topics from the first 4 bytes of the SHA3 of each topic.
- Set user-given attribute ttl (time-to-live).
- Set the expiry as the present Unix time plus the time-to-live.
- Set the nonce as that which provides the most work proved as per the previous definition, after some fixed amount of time of cycling through candidates or after a candidate surpasses some boundary; either should be given by the user.
- Finally the message can be sent over the Ethereum blockchain using ssh.post

3 Related Work

There has been numerous protocols for P2P communication:

- UDP [2]: Datagram messaging systems. Simple connectionless transmission model. No guarantee of delivery, ordering, security, or duplicate protection.
- 0MQ [3]: A distributed messaging system, no inherent privacy safeguards.
- Bitmessage [4]: Decentralized, encrypted, P2P, trustless communications protocol. Using public-key cryptography. Inefficient use of resources.
- TeleHash [5]: An encrypted mesh protocol, adopted in the IBM-Samsung's ADEPT proof-of-concept. Similar in approach to BitTorrent. Use DHT (Distributed Hash Table) to do deterministic routing.
- Tox [6]: Also use DHT but vastly simplified packet format and encryption.

None of these protocols provides a completely "dark" messaging (Dark means there is no reliable methods for tracing packets). Whisper combines aspects of both DHTs and datagram messaging systems (e.g. UDP) to ensure both security and performance. In addition, for a home system, most of the messages across the network do not require storage, thus Whisper can be applied here. We also found that Ethereum claimed to confirm a message/transaction in a fraction of a second, which encouraged us to experiment with Ethereum for the low latency. Unlike Bitcoin network, every transaction in Ethereum requires a fee to process that increases as more bytes are put into the message. This is a factor that we will also test for: to see whether whispering requires fees to process, how much the fee will be for each message, whether total messaging cost is scalable for a home system.

4 Current Status and Plan

Currently we have read through the Ethereumj repository on Github and the documentation of the Whisper protocol (the protocol itself is included in the Ethereum implementation). Our plan would be to create Whisper clients on two Androids phones and send messages between them by April. After we set up the communication link, we plan to finish data collection, format, and processing by May. The rest of the experiment is integrating two parts together, run the experiment and do time-cost analysis to evaluate the feasibility of the protocol.

References

[1] Ethereumj,
`http://ethereumj.io`

[2] User Datagram Protocol,
`https://en.wikipedia.org/wiki/User_Datagram_Protocol`

[3] Distributed Messaging - zeromq,
`http://zeromq.org`

[4] Bitmessage,
`https://bitmessage.org/wiki/Main_Page`

[5] Telehash - encrypted mesh protocol,
`http://telehash.org`

[6] Tox,
`https://github.com/irungentoo/toxcore`