Anchor: An NDN-Based Blockchain Network

Shucheng Yu[1(✉)], Noor Ahmed[2], and Ruiran Wang[1]

[1] ECE Department, Stevens Institute of Technology, Hoboken, NJ 07030, USA
{shucheng.yu,ruiran.wang}@stevens.edu
[2] Air Force Research Laboratory, Rome, NY 13441, USA
norman.ahmed@us.af.mil

Abstract. Efficient broadcasting is critical for timely propagation of transactions and blocks to the network in blockchain systems. Existing blockchain networks are usually characterized by a high communication overhead and redundant traffics which adversely impact the transaction/block latency and security. In the networking community, remarkable progresses have been made in past decades on network efficiency. In particular, information-centric networking (ICN) has been considered a promising approach for network efficiency and robustness by shifting from traditional host-centric paradigm to the content-centric paradigm. However, direct marriage between blockchain and ICN does not yield a satisfying solution due to various security gaps. In this paper, we make the first attempt to integrate the two by introducing a new blockchain network technique called *Anchor* based on an ICN technique - named-data networking (NDN). By tailoring NDN and cascading two NDN systems, we demonstrate an efficient yet secure solution for data propagation in blockchain systems. Simulation shows that Anchor consumes less bandwidth and has a lower transaction/block latency as compared to the Bitcoin flooding network and a recent technique Erlay.

Keywords: Blockchain · Information-centric network · Named-data network · Efficiency · Security · Privacy

1 Introduction

In blockchain systems it is crucial to timely propagate data (transactions or blocks) so as to achieve a consistent view and consensus among distributed participants (e.g., miners in Bitcoin). An efficient network layer is not only indispensable to blockchain performance including throughput, robustness, scalability and bandwidth efficiency. It is also closely relevant to the security of blockchain. Slow data propagation results in long synchronization delay, which can lead to

Effort sponsored by the Air Force under MOU FA8750-15-3-6000. The U.S. Government is authorized to reproduce and distribute copies for Governmental purposes notwithstanding any copyright or other restrictive legends.

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 3–9, 2021.
https://doi.org/10.1007/978-3-030-80851-8_1

security vulnerabilities [1,2] including blockchain forks [3]. Inappropriate network layer design may also lead to the eclipse attack [4] wherein attackers dominate the in- and out-bound communications of a victim.

In the popular permissionless blockchain Bitcoin, the gossip protocol is used for data propagation and consensus is achieved by "proof of work" (PoW). In the gossip protocol, data (transactions/blocks) is relayed by mining nodes in a greedy way - each node receiving the data immediately propagates to its neighbors by exchanging three messages - *inv* (announcement), *getdata* or *getblocks* (request) and *tx* or *block* (delivery of data). While the three-way message exchange can reduce redundant transmissions of actual data (transactions/blocks), it incurs a significant management overhead because of the announcement and request messages. Existing study [6] shows that announcement messages can count for around 30–50% of overall traffic of which 88% are redundant. Moreover, the per-hop three-way message exchange also causes additional delays as compared to direct unsolicited "push" of the data. Permissioned blockchain systems [5] are mostly based on the classical Byzantine fault tolerant (BFT) protocol or its variants for consensus. The BFT-based consensus usually involves multi-round broadcastings which share similar properties as the gossip protocol.

Existing techniques for blockchain network efficiency mainly resort to three approaches - deploying relay nodes to suppress redundant traffics [7], compressing transactions to reduce the message sizes [8], or limiting the fanout of forwarding links at each node to control the degree of redundancy [6]. However, these techniques either assumes centralized relay nodes are deployed [7], which is incompatible with the decentralized nature of blockchain, or limits the fanout of forward links [6], which inevitably prolongs the message propagation time and cause a larger delay to each transaction. Compressing transactions is an orthogonal technique that can be integrated to other designs including this work.

In the network community, remarkable progresses have been made in past decades on network efficiency. In particular, information-centric networking (ICN) [9] has been considered a promising approach for network efficiency and robustness by shifting from traditional host-centric paradigm to the content-centric paradigm. Essentially, the gossip protocol can be considered a special instance of ICN in the sense that data is published and requested by content. One difference is that ICN supports both asynchronous and synchronous production and consumption of data while the gossip protocol works in the synchronous manner. As compared to the synchronous counterpart, asynchronous production and consumption provides more flexibilities. For example, consumers can subscribe data in advance before it is produced. Once available, data can be directly pushed to consumers, saving the real-time announcements and requests. However, direct application of the asynchronous mode in blockchain faces non-trivial challenges. Specifically, as the content of data and even its producer are unknown before the data is produced, consumers do not know what and where to prescribe. One solution is to use designated relay nodes as in existing work [7]. However, this will require the relay nodes to be trusted. Yet another problem is that the "prior-subscription" through the designated relay nodes may reveal traffic pattern and lead to security vulnerabilities.

In this paper, we introduce a new blockchain data propagation protocol namely *Anchor* by tailoring an ICN technique - named-data networking (NDN) [10]. Specifically, we design a cascaded two-layer tailored NDN for asynchronous transaction/block production and consumption. Instead of using designated trusted relay nodes, we allow periodical election of a set of random relay nodes which we call *anchor nodes* (or *anchors*). While multiple random anchors improve system robustness under traffic manipulation attacks, redundant traffics from multiple anchors can be aggregated in our design. We protect traffic privacy by randomizing data propagation routes of each anchor node. Simulation with NDNSim shows that our design achieves a lower communication overhead and a shorter propagation latency as compared to Bitcoin gossip protocol and Erlay.

2 Our Design

The overall idea of our design is to periodically elect a set of random anchor nodes to relay data to the network. The asynchronous data production and consumption is achieved through a cascaded two-layer NDN: in layer 1, anchors act as consumers and each other node as a potential producer; in layer 2, each anchor node acts as a producer and all other nodes as consumers. Newly produced data will first propagate through the layer-one NDN to anchors, which relay to other nodes over layer-two NDN. To allow prior-prescription before data is actually produced, we use time slots t_i, $i = 1, 2, \cdots$, as the "names" for data. In other words, all data produced within the same time slot t_i will be propagated through pre-determined routes. To protect traffic privacy, each node subscribes from a random up-steam node for each time slot t_i. Therefore, the actual routes in t_i is randomly and distributedly selected by all nodes en route. The level of privacy can be further adjusted by selecting an appropriate duration for time slots t_i. In our design, redundant transmissions of data relayed by different anchor nodes are detected and aggregated at each node en route they first reach.

2.1 Random Anchor Selection

Anchors are randomly selected at every interval T ($T \geq t_i$). As there is no central party in the system, the selection shall be in fully distributed manner. For this purpose, we utilize the recently confirmed blocks as common seeds. Assume $\{B_{t-k}, B_{t-k+1}, \cdots, B_{t-1}, B_t\}$ are the $k+1$ recently confirmed blocks at time t. The anchor node is selected by a selector $i = h(h(B_{t-i})|h(B_t)|tstmp)\ mod\ N$, where N is the approximate total number of nodes in the system, $h()$ a one-way hash function, and $tstmp$ the timestamp associated with B_t. Due to the unpredictability of blocks, the number i is difficult to predict before the latest block is confirmed. To automate the node selection in the distributed way, a node j is selected as an anchor node only if $i == Encrypt_{sk_j}(h(B_t)|tstmp)\ mod\ N$,

where sk_j is the private key of node j. Given the randomness of the encryption function $Encrypt()$, the probability that the equality holds is $1/N$ for a random node j and on average one node is selected by the selector i. And k selectors will identify k random anchors on average. Note that, in case more than one node are selected by the same selector, we accept all of them as anchor nodes. The anchor nodes can be updated at every interval T by changing the selector i as $h(h(B_{t-i})|h(B_t)|tstmp + T)\ mod\ N$.

Once a new block is confirmed, each node first checks if it is an anchor node by locally calculating the k selectors and then compare if $Encrypt_{sk_j}(h(B_t)|tstmp)\ mod\ N$ matches with any selector. If not, this node is not an anchor. Otherwise, it broadcasts the tuple $ancmt = (Encrypt_{sk_j}(h(B_t)|tstmp), i, pk_j, idx_j)$ to its neighbors as the announcement. Each node receiving the announcement verifies whether 1) $i == h(Decrypt_{pk_j}(Encrypt_{sk_j}(h(B_t)|tstmp)))\ mod\ N$ and 2) the decrypted $h(B_t)||tstmp$ matches with the block B_t in its local blockchain, where pk_j is the sender j's public key and i the selector included in $ancmt$. To thwart anchor manipulation, we restrict pk_j to those previously appeared in a confirmed block (e.g., as an address in a transaction) indexed by idx_j. This prevent attackers from temporarily generating a private key (e.g., by exhaustive search) in order to be selected as the anchor. The random distribution and unpredictability of anchors also help thwart traffic manipulation related attacks.

One potential problem may occur when there are forks with the blockchain. This may cause some nodes refuse to accept some valid anchors because they do not share the same longest chain of blocks. This can be addressed by accepting all $ancmt$ messages with valid selector i (via step 1) but letting anchors to include the hash values and nonces of p previous blocks in $ancmt$. Each node can verify the authenticity of $ancmt$ messages by checking their respective proofs of work (without payload).

2.2 Two-Layer Cascaded NDN Design

To facilitate efficient propagation of transactions and blocks by aggregating announcement and request messages in the gossip protocol, we design a two-layer cascaded NDN network as shown in Fig. 1. The broadcasted tuple $ancmt$ will serve as the announcement for both NDN networks within current interval T.

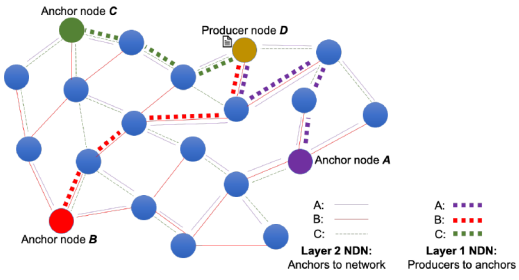

Fig. 1. Two-layer cascaded NDN

At layer 1, the data producers and the anchors form an NDN network. In a blockchain system, any node can be a potential data producer. Therefore, each anchor shall subscribe to every other node for all new transactions or blocks. As the subscription in layer 1 NDN is opposite to the direction of the propaga-

tion of *ancmt*, we let each node create layer-1 pending interest table (PIT) after layer 2 PIT has been created as discussed below.

In layer 2 NDN network, each anchor node acts as a producer and other nodes are consumers. The construction of the PIT table of layer 2 NDN network is straightforward: on receiving the announcement message (i.e., the verified *ancmt*), each node n_i randomly selects a sender n_j from all up-steam neighbors who relayed the *ancmt* message and sends an interest packet for time slot t_j to it for each future time slot $t_j \in T$. On receiving the interest packet, n_j creates a PIT entry with n_i as the outcoming interface for t_j. Meanwhile, n_i creates a PIT entry for layer 1 NDN for t_j with n_j as the outcoming interface. This means that n_i will forward all new data generated in t_j to n_j. In the layer 2 PIT tables, each entry also includes the index of the anchor, meaning that PIT entry is for that particular anchor node at time t_j.

Please note that in our design, the routes (i.e., PIT entries) of the nodes form a random broadcast tree rooted at each anchor for each time slot t_j. Layer 1 NDN also forms a random broadcast tree rooted at each node with all anchors being leaf nodes. A branch in a layer 1 NDN broadcast tree overlaps with a branch in a layer 2 NDN broadcast tree but with an opposite data propagation direction. Traffic pattern privacy is also well protected in our design. This is because for each time slot t_j, broadcast trees of both layer 1 and layer 2 are randomly determined by each node hop by hop.

2.3 Elimination of Duplicated Transmissions from Multiple Anchors

Another problem we need to address is the duplicated data traffics relayed by multiple anchors. This is because when a new data is generated, the producer broadcasts it to all anchors, each of whom relays the data to the entire network.

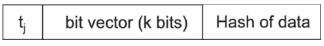

Fig. 2. Short message of duplicated data

To reduce duplicated transmissions, we design a short message for duplicated data as shown in Fig. 2. The bit vector assigns one bit for each anchor and is initialized as all 0's. Bit 1 means the data has been received from that anchor. When a node receives a data, it first updates its local bit vector by ORing its local bit vector with the received one. Then it checks its PIT entries. For those already fulfilled (i.e., data has been forwarded to the outcoming interface), it will just send the short message as shown in Fig. 2 (the hash of data can be replaced with a shorter index defined by each anchor); for those not fulfilled, the node will send the full message, which includes t_j, the updated bit vector, and the original data. In this way, duplicated transmissions are aggregated into short messages. We can apply encoding techniques to further compress the bit vector.

3 Performance Evaluation

We evaluate our design using NDNSim with 60,000 nodes, each connected to an average of eight other nodes. Transaction generation rate is seven per second.

T is 4 min and t_j is 1 min. We compare Anchor with the Bitcoin flooding network and Erlay for transaction propagation regarding bandwidth consumption and per-transaction latency.

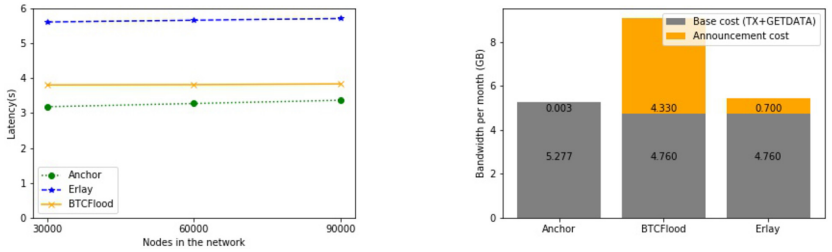

Fig. 3. Comparison of Anchor with BTCFlood and Erlay

As shown in Fig. 3, both the latency and the overall bandwidth consumption of Anchor are smaller than both Erlay and Bitcoin. The latency of Anchor increases slightly faster than the other two as the network size increases. This is because we used a fixed number of 40 anchor nodes, which become sparser when the network size increases. Although this increased latency can be flattened by deploying more anchor nodes, it will introduce more communication overhead because of redundant transmissions of anchors. As shown in the right figure, Anchor is able to aggregate almost all announcements (0.003 GB) within each time interval T. But it has a slightly higher overhead with the base cost (5.277 GB) mainly contributed by the bit vector.

4 Conclusion

In this paper, we introduce the first NDN-based efficient network layer for Bitcoin-like blockchain systems. With the new two-layer cascaded NDN design, we are able to significant aggregate the management messages of the gossip protocol with traffic pattern protected. Extensive simulation with a widely used simulator NDNSim shows that our design enjoys a lower communication overhead and a smaller latency as compared to the state of the art.

References

1. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
2. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 291–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_10

3. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: IEEE P2P, Italy, Trento, pp. 1–10 (2013)
4. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: USENIX Security 2015, Washington D.C., USA, pp. 129–144 (2015)
5. Xiao, Y., Zhang, N., Lou, W., Hou, Y.T.: A survey of distributed consensus protocols for blockchain networks. IEEE Commun. Surv. Tutor. **22**(2), 1432–1465 (2020)
6. Naumenko, G., Maxwell, G., Wuille, P.: Erlay: efficient transaction relay for bitcoin. In: ACM CCS 2019, London, UK, pp. 817–831 (2019)
7. Falcon. https://www.falcon-net.org/
8. Compact block relay. https://github.com/bitcoin/bips/
9. Information-Centric Networking. https://irtf.org/icnrg
10. Named-Data Networking. https://named-data.net/