Throughput Optimal Routing in Blockchain Based Payment Systems

Sushil Mahavir Varma
sushil@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Siva Theja Maguluri
siva.theja@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

ABSTRACT

Cryptocurrency networks such as Bitcoin have emerged as a distributed alternative to traditional centralized financial transaction networks. However, there are major challenges in scaling up the throughput of such networks. Lightning network [28] and Spider network [32] are alternates that build bidirectional payment channels on top of cryptocurrency networks using smart contracts, to enable fast transactions that bypass the Blockchain.

In this paper, we study the problem of routing transactions in such a payment processing network. We first propose a Stochastic model to study such a system, as opposed to a fluid model that is studied in the literature. Each link in such a model is a two-sided queue, and unlike classical queues, such queues are not stable unless there is an external control. We propose a notion of stability for the payment processing network consisting of such two-sided queues using the notion of on-chain rebalancing. We then characterize the capacity region and propose a throughput optimal algorithm that stabilizes the system under any load within the capacity region. The stochastic model enables us to study closed loop policies, which typically have better queuing/delay performance than the open loop policies (or static split rules) studied in the literature. We investigate this through simulations.

KEYWORDS

Routing, Payment processing networks, Blockchain, Two Sided Queues, MaxWeight, Dual Descent, State Dependent, Foster-Lyapunov Theorem

ACM Reference Format:
Sushil Mahavir Varma and Siva Theja Maguluri. 2020. Throughput Optimal Routing in Blockchain Based Payment Systems. In *ACM Mobihoc: International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, June 30th - July 3rd, 2020, Shanghai, China.*

1 INTRODUCTION

Blockchain is a decentralized, distributed public ledger that is secured based on consensus and cryptographic hashing, and was first introduced in the seminal paper [25] to build the Bitcoin cryptocurrency Systems. Over the last decade, Blockchain has been used in a variety of applications, including to build other cryptocurrency networks [40], Smart Contracts [7, 10], networks for financial services [15], supply chain management [26] etc. In addition to the public blockchains such as Bitcoin and Ethereum, several companies are building private Blockchains [22] and are also involved in Hybrid Blockchains [42].

This paper studies payment transaction networks. In a centralized transaction network, a centralized body such as Paypal or Visa is responsible for facilitating all the transactions between any pair of sender and receiver. It is an efficient, fast and established payment system which achieves processing rate of thousand of transactions per second with only few seconds of delay. In such a network, each user of the system, has an account or some relation with the payment processor (possibly through another entity such a bank). In contrast, in a decentralized transaction network, no such centralized entity is required. Credit networks such as [14] and Blockchain based public cryptocurrency network such as Bitcoin [25] or Ethereum [40] or a private network such as Ripple [4] are examples of decentralized transaction network. The focus of this paper is on Blockchain based payment networks even though some of the results can be applied to other payment processing networks such as credit networks. Blockchain based payment networks have the potential of being a more secure and private way of transacting and promise significantly cheaper transaction costs by eliminating the premium paid to the central authority for each transaction. However, Scalability is a major challenge in such networks. Visa processes 1667 transactions per second [39]. On the other hand, Bitcoin, is limited to about 3-7 transactions per second [39] due to the underlying Proof of Work paradigm that secures Bitcoin. Moreover, Proof of Work involves computing cryptographic hashes, called mining which is not only expensive, but incurs huge energy costs.

Solutions such as Proof of Stake [20] are proposed to address scalability issues in Blockchain [9] and this is an area of active research. One way of addressing the scalability issues for the purpose of processing payments is to build a second payment processing networks on top of a Blockchain such as Bitcoin. Most of the transactions are processed in this second network, while the Blockchain with its expensive mining is used sparingly. Such a payment network is composed of bidirectional peer to peer payment channels, which are secured using smart contracts. Examples of such payment processing networks include Lightning network for Bitcoin [28], Raiden network for Ethereum [23], and Atomic swap for inter Blockchain transactions [18]. The focus of this paper is such Blockchain based secondary payment processing networks. Routing algorithms for such payment processing networks are studied in Silent Whispers [24] and Speedy Murmurs [29].

ACM MobiHoc'20, June 30th - July 3rd, 2020, Shanghai, China

The essential idea behind such networks is illustrated by the following example. Two agents, say Alice and Bob create a channel with capacity c\$ by each buying Bitcoins (or equivalently, recording a transaction in the underlying Blockchain) worth c\$. One may think of this channel as a kind of joint account worth $2c$\$ between Alice and Bob. Whenever, Alice wants to send Bob $a < c$\$, they add a new smart contract adjusting their individual ownership in the $(2c)$\$ Bitcoins, so that Alice owns $(c - a)$\$ and Bob owns $(c + a)$\$. The smart contract protects the transaction between Alice and Bob. Later, if Bob wants to send Alice some money, a new smart contract will be added reflecting effective ownership. However, due to the limit on the channel capacity, they can only send at most c\$ in each direction in a time slot. Each transaction takes some finite time, which defines the duration of a time slot. Moreover, the difference in total amount money sent in the two directions can be at most c\$. If Alice and Bob each want to send the other $b > c$\$, they first send each other c\$ in one time slot and buffer the remaining amount. At the end of one time slot, the channel is balanced and the buffered requests can be sent in the following time slots. In this approach, the underlying Bitcoin (Blockchain) network is used only to create a payment channel or to close a payment channel. All the other transactions are processed in the second overlay networks using smart contracts.

Now suppose that, in addition, Bob and Chris have a channel between them, but Alice and Chris don't. Suppose Alice wants to send payments to Chris, Alice could send them to Bob, and Bob could forward them to Chris. Bob will be paid a premium to help process the transaction. In general, a transaction can be processed by finding a path between the sender and the receiver in the payment graph, where the agents are vertices and the channels are edges. All these transactions do not need to go through the underlying Blockchain, and so this is known as off-chain rebalancing. However, note that this works only if on average, the requests in each direction on a link are balanced. Suppose there is a large amount of buffered requests from Alice to Bob, and no outstanding requests from Bob to Alice. These requests can only be met when there are corresponding future requests from Bob to Alice. One can then send some payment through the underlying Bitcoin network (Blockchain) to reduce the 'imbalance' on this channel. Such a transaction is called on-chain rebalancing, and is usually more expensive because it involves cryptographic mining.

There are several issues in the design of such payment networks, such as privacy, security, efficiency etc [24] [29] [32]. The focus of this paper is in designing a path from the sender to the receiver of each transaction so that the overall throughput of the network is maximized while limiting the use of on-chain rebalancing. While the routing problem is studied in the literature [24, 29, 32], a fluid model was used and a static scheduling policy is proposed. While static policies that do not use the system state while making decisions can maximize throughput, they are known to be sub-optimal in terms of other metrics such as queue lengths and delays. In this paper, we propose a stochastic model of the payment processing network that keeps track of the buffered requests and the imbalance on the channels.

When a transaction is processed using off chain rebalancing on a channel, the individual ownership changes, and needs to be updated. Since we are interested in the long term throughput behaviour of the system, the system is sustainable only if on average, equal number of payments are made in both directions of each channel. Therefore, we make the simplifying assumption that at any time, on each channel, a request is served in one side, if and only if a request of equal value is served in the opposite direction. For example, if Alice wants to send a\$ to Bob and Bob wants to send b\$ to Alice and c\$ is the capacity of their channel, then, $\min\{a, b, c\}$\$ is served in both directions and the remaining request is buffered. The requests are buffered until there is enough demand in the opposite direction or there is enough capacity on the channel. This ensures that we don't have to constantly update the ownership of the channel, and makes the analysis simpler. The channels can thus be naturally modeled by a two-sided queue.

Unlike traditional queues, where a server is fixed, and serves arriving demand, in a two-sided queue, both requests and servers arrive and are queued up. Each request is paired up with a server and both instantaneously depart from the system at the finish of service. Note that the 'servers' in this context also correspond to requests in the payment network, since there is no conceptual distinction between the arrivals on the two sides of a two-sides queue. Even though there are various notions of stability, positive recurrence of the underlying Markov Chain is a strong form that ensures the existence of a steady-state distribution, and enables one to study the stationary queue lengths, delays, mean imbalance etc. Even if the arrival rates in both directions are equal, a two-sided queue is only null-recurrent and not positive recurrent. We need an external control to make it positive recurrent. Therefore, we use on-chain rebalancing to define stability based on positive recurrence in such a system.

Main Contributions: The main contributions of this paper are listed below.

(1) We propose a Stochastic Model to study routing of requests in a Blockchain based payment processing networks. Such a Stochastic Model enables the study of performance of state dependant or closed loop routing policies.
(2) We model such a payment processing network as a network of two-sided queues, and define stability of the network based on positive recurrence using the notion of on-chain rebalancing. We then characterize the capacity region of the payment processing network consisting of all demand rates under which the system can be made stable.
(3) We propose a novel state dependant policy and prove that it is throughput optimal, i.e., it stabilizes the system under any arrival rate in the capacity region. Since this is a stronger notion of stability than the rate stability studied in [11], one can study steady-state behaviour of various metrics of interest.
(4) Under our proposed routing policy, we obtain an upper bound on the total buffered payment requests, and discuss its trade off with the amount of on chain rebalancing.
(5) We study the performance of the proposed algorithm using simulation on Ripple network data-set. The proposed algorithm routes $\sim 65\%$ of the incoming demand using only off-chain rebalancing and we note that it needs lesser amount of on-chain rebalancing than Spider Network [32].

Notation: Now, we introduce some general notation used in this paper. We denote real numbers by $\mathbb{R}$ and non negative real numbers by $\mathbb{R}_+$. Integers are denoted by $\mathbb{Z}$, whereas the non negative integers are denoted by $\mathbb{Z}_+$. We denote $\max\{0, A\}$ by $[A]^+$ and indicator function denoted by $\mathbb{1}_A$ is 1 if A is true or 0 otherwise. For a collection of scalar variables x indexed from some set, we use a bold $\mathbf{x}$ to denote a vector containing all its components.

1.1 Organization

The rest of the paper is organized as follows: In Section 2, we present an overview of related work. In Section 3, we introduce the model, assumptions and key notations. We also define our notion of stability in this section. In Section 4 we propose a routing algorithm and prove that it is throughput optimal using the Foster-Lyapunov theorem. Next, we consider the modified routing algorithm in section 5. We present the simulations in Section 6 before concluding in Section 7.

2 RELATED WORK

In this section, we present the prior work on payment processing networks, two-sided queues and MaxWeight family of algorithms.

Lightning network is a peer-to-peer path based transaction network built on top of Bitcoin [28]. It introduces a privacy preserving method to process transactions between two users via possibly routing it through multiple users. It however, does not provide a method to choose a path between a sender and receiver. Then, [41] presents a privacy preserving algorithm to find a shortest path between two nodes in a centralized manner. Later, SilentWhispers [24] presents a privacy preserving protocol to route transactions in a decentralized manner using the idea of landmark routing based credit network design [36]. In short, SilentWhispers is a routing, payment and an accountability algorithm in decentralized transactions routing to ensure privacy. It was the first distributed, privacy-preserving credit network. Speedy Murmurs [29] is a more efficient routing algorithm which guarantees similar privacy levels. After that, [32] looked at routing cryptocurrency with the Spider network. Their simulation results show that Spider improves the volume and number of successful payments substantially comparing to the state-of-the-art.

Two-sided queues are studied in a variety of different contexts in the literature. Since the notion of stability in two-sided queues is not straight forward and one needs external control to achieve positive recurrence, several different approaches are taken in the literature. Spider network [32] which studied a fluid model of the payment processing network is the closest work in the literature. Rate stability was used there to bypass the issues about positive recurrence. One limitation of this approach is that rate stability does not ensure the existence of a steady-state distribution. A general framework of matching queues that subsumes two sided queues was studies in [16]. But the focus in [16] is to minimize the queue lengths over a finite time-horizon, and not longer term throughput.

Two-sided queues naturally arise in ride-hailing systems, where riders are the requests and drivers are the servers. Closed queuing network models were used in [5] and [6] to study ride-hailing systems when the total number of cars (servers) in the system are fixed. Issues about recurrence are avoided by dropping customer demand to make sure that the rider queues does not blow up to infinity. External control in terms of on-demand servers was used in [27] to make the system stable. Caldentey et al. [8] and Adan and Weiss [1] looked at a simplifying model which can be thought of as allowing customer arrivals only when there are servers waiting, and show positive recurrence under this model. Two sided queues for Kidney exchanges were studied in [31] [3] and for barter exchanges with dynamic matching were studied in [2].

Tassiulas and Ephremides [35] proposed the celebrated MaxWeight algorithm for scheduling in downlink in mobile base stations and its generalization, the Backpressure algorithm for routing and scheduling in multihop wireless networks [35]. This then led to a huge line of literature on resource allocation problems in wired and wireless networks and the book [33], presents an excellent exposition. MaxWeight family of algorithms arise naturally when one views the scheduling problem as a fluid-like optimization problem, and consider a gradient descent algorithm on its dual. However, unlike open-loop policies that one obtains in the fluid limit, here, one obtains a closed loop or state dependant policies that are shown to maximize throughput using Foster-Lyapunov Theorem. In addition to networking problems, these ideas have been used in other resource allocation problems such as Cloud computing [21], Online ad matching [34], ride sharing [19] etc.

3 MODEL AND DEFINITIONS

In this section, we first present the model and then the definitions of stability. We will also define the capacity region of the system and show that no family of algorithm can make the system stable for the demand rate outside the capacity region.

3.1 The Model

We consider a payment processing network consisting of payment channels based on a Blockchain such as Bitcoin or other cryptocurrency systems. We model the payment processing network by a payment graph, $G(V, E)$, where, the vertices V are the agents in the network and the edges E are the payment channels between the agents. We assume that the graph G is connected. All the edges in this graph are bidirectional. This means that we have a directed graph such that if $(u, v) \in E$, then, $(v, u) \in E$. Each edge $(u, v) \in E$ has a capacity $c_{(u,v)}$ such that $c_{(u,v)} = c_{(v,u)}$. We consider this for the ease of exposition and our results can be easily extended for unequal capacities and unidirectional edges. Also define $c_{max} = \max_{(u,v)\in E} c_{(u,v)}$. We consider a discrete time model and let $a_{ij}(t)$ denote the value of payment request that arrives at time t and should be sent from agent i to agent j. The demand $a_{ij}(t)$ is assumed to be in $\mathbb{Z}_+$ since it may be expressed in terms of smallest possible denomination such as cents, Satoshi for Bitcoin [37] and Wei for Ethereum [40]. The arrivals are assumed to be an iid sequence of random variables with mean λ_{ij} and finite variance. Moreover, $a_{ij}(t)$ are assumed to be independent for different $i - j$ pairs.

Each edge (u, v) in the graph G is a bidirectional payment channel between agents u and v built on top of a cryptocurrency system such as the Blockchain. Each edge (u, v) is modeled as a two-sided queue consisting of two buffers of outstanding demand from u to v and v to u. The value of outstanding payment requests in these buffers is denoted by $q_{(u,v)}$ and $q_{(v,u)}$ respectively.

The focus of this paper is on designing a routing algorithm. For each payment request that arrives from i to j, the goal is to find one or more paths from i to j in the graph G on which this payment will be sent. We denote by P_{ij} the set of all the possible paths from i to j. Since we assume that the graph is connected, the set P_{ij} is nonempty for all $i \neq j \in V$. Each element in P_{ij} is a set of non repeated edges which connects i to j. Demand arising between i and j can be met using multiple paths in the set P_{ij}. We assume that the set of valid paths P_{ij} is given. This can be the set of all possible paths from i to j in the case of decentralized path based transaction networks such as [32]. In partially centralized networks such as [24], the set P_{ij} is restricted to the set of paths that pass through landmark nodes. Let $\mathcal{P} = \cup_{i\neq j\in V} P_{ij}$ denote the set of all valid paths.

The routing algorithm determines the amount of payment requests $x_p(t)$ to be routed through each path $p \in P_{ij}$. Note that the assignments $x_p(t)$ should be picked such that $\sum_{p\in P_{ij}} x_p(t) = a_{ij}(t)$ for all $i, j \in V$, so that all the arriving requests are assigned to some path. Once the amount of payments $x_p(t)$ are determined for each path $p \in \mathcal{P}$, a request of $x_p(t)$ is added to all the buffers on the path p. We use $y_{(u,v)}(t)$ to denote the total amount of payment request that is added on the channel (u, v) at time t, i.e.,

$$y_{(u,v)}(t) = \sum_{p\in\mathcal{P}:(u,v)\in p} x_p(t). \tag{1}$$

At each time t, on each edge (u, v), the amount of service is the maximum possible values up to the channel capacity $c_{(u,v)} = c_{(v,u)}$, such that an equal amount of requests on either side of each channel are served. If $s_{(u,v)}(t)$ denotes the amount of payment requests served in buffer $q_{(u,v)}$, then, we have

$$s_{(u,v)}(t) = \min\left\{q_{(u,v)}(t), q_{(v,u)}(t), c_{(u,v)}\right\}. \tag{2}$$

Note that by the symmetry in the definition, we have that $s_{(u,v)}(t) = s_{(v,u)}(t)$.

In addition to routing, a second control decision that needs to be made is about on-chain rebalancing. Recall that most of the requests are served through smart contracts in the payment processing networks, and bypass the use of the underlying Blockchain and is called off-chain rebalancing. However, some of the demand may sometimes be served by going through the Blockchain, and this is called on-chain rebalancing, and is more expensive. The second control decision is to decide how much of the outstanding demand in each direction on each channel is met using on-chain rebalancing. The goal is to minimize the use of on-chain rebalancing. Let $r_{(u,v)}(t)$ denote the amount of on-chain rebalancing on the link (u, v) at time t. The outstanding demand in each buffer evolves as

$$q_{(u,v)}(t+1) = q_{(u,v)}(t) + y_{(u,v)}(t) - s_{(u,v)}(t) - r_{(u,v)}(t). \tag{3}$$

We assume that the amount of on-chain rebalancing satisfies $r_{u,v}(t) \leq q_{(u,v)}(t) + y_{(u,v)}(t) - s_{(u,v)}(t)$ so that the outstanding requests, $q_{(u,v)}(t+1)$ do not go negative. Even though we say routing algorithm, we consider algorithms that perform routing as well as on-chain rebalancing. For a given routing algorithm, we say that the rate of on-chain rebalancing is at most ϵ if the long term average amount of on-chain rebalancing on all links is smaller than ϵ with probability 1, i.e., $\limsup_{T\to\infty}\left(\sum_{t=1}^{T}\sum_{(u,v)} r_{u,v}(t)\right)/T \leq \epsilon$ w.p. 1.

3.2 Stability and Capacity Region

Consider a classical queue, where server serves requests at rate μ and the requests arrive according to some stochastic process at rate λ. As long as the arrival rate is smaller than the service rate, i.e., $\lambda < \mu$ the system is usually stable, for various different notions of stability.

However, the situation in a two-sided queue is more involved. We now have two queues, and requests in both the queues are paired for service. As an example, consider a payment network consisting of a single channel (u, v) with infinite capacity, and iid Bernoulli payment requests on both sides. This is a Discrete Time Markov Chain (DTMC). If the rate of arrivals on each side is different, i.e., if $\lambda_{(u,v)} > \lambda_{(v,u)}$ or $\lambda_{(u,v)} < \lambda_{(v,u)}$, then the system is transient and so not stable. When $\lambda_{(u,v)} = \lambda_{(v,u)}$, the two-sided queue is said to be rate-stable. However, the underlying DTMC is still not very well-behaved, because it is null-recurrent since it is a symmetric random walk in one dimension. In particular, there is no steady-state distribution, and the limiting buffer lengths may not be bounded, i.e, $\lim_{C\to\infty}\lim_{t\to\infty} P(q_{(u,v)}(t)+q_{(v,u)}(t) \geq C) \neq 0$. Null-recurrence is not a consistent notion of stability. This is because of the following: now consider a network consisting of three channels similar to (u, v) that are not connected to each other, and so are independent. This system is equivalent to a symmetric random walk in three dimensions, which is transient. Therefore, we cannot take null-recurrence as a notion of stability.

In this paper, we will consider two notions of stability, that are both stronger than rate stability that are studied in literature [12]. One is in terms of positive recurrence of the DTMC, and the other is that $\lim_{C\to\infty}\lim_{t\to\infty} P(q_{(u,v)}(t)+q_{(v,u)}(t) \geq C) = 0$. The previous example shows that without external control, a two-sided queue is not stable under either of these notions. In other words, off-chain rebalancing alone cannot make the payment network stable. So, we need at least some on-chain rebalancing to make the system stable. However, it turns out that an arbitrarily small amount of on-chain rebalancing is enough. Therefore, instead of studying routing algorithms, we study families of algorithms, where for any given $\epsilon > 0$, each family consists an algorithm with on-chain rebalancing rate smaller than ϵ. We first present the weaker notion of stability.

Definition 3.1 (Weak Stability). Consider the payment processing network with a given arrival rate vector $\boldsymbol{\lambda}$. A family of routing algorithms is said to weakly stabilize the network if for any $\epsilon > 0$, there exists an algorithm in the family that uses at most ϵ rate of on-chain rebalancing such that, $\lim_{D\to\infty}\lim_{t\to\infty}\mathbb{P}(\sum_{(u,v)\in E} q_{(u,v)}(t) \geq D) = 0$.

In classical single sided queuing systems, weak stability is defined as the set of arrival rate vectors for which there exists an algorithm under which we have $\lim_{D\to\infty}\lim_{t\to\infty}\mathbb{P}(\sum_{(u,v)\in E} q_{(u,v)}(t) \geq D) = 0$. Although, in our case we need to consider a family of algorithms to define weak stability due to the following two reasons:

- With zero on chain rebalancing, it is not possible to stabilize the system as discussed above.
- With arbitrary on chain rebalancing, it is meaningless to talk about stabilizing the system as after routing incoming demands using any algorithm, the excess demand can be

routed using on-chain rebalancing and the system can be stabilized under *any* algorithm.

Thus, we need a *family* of algorithm such that the system is stabilized under any given constraint on on-chain rebalancing. Based on our notion of stability, we now define the capacity region.

Definition 3.2 (Capacity Region). The capacity region for the payment processing networks is the set of arrival rate vectors $\boldsymbol{\lambda}$ that are weakly stabilizable by some family of routing algorithms.

We are interested in families of algorithms that stabilize the network for the maximum possible set of arrival rates, and so we now define throughput optimality.

Definition 3.3 (Throughput Optimality). A family of algorithms is said to be throughput optimal if it can weakly stabilize the payment processing network under any arrival rate in the capacity region.

Let C define the set of arrival rates $\boldsymbol{\lambda}$ if there exists an $\mathbf{x} \in \mathbb{R}_+^{|\mathcal{P}|}$ such that,

$$\sum_{p \in P_{ij}} x_p = \lambda_{ij} \; \forall i \neq j \in V, \tag{4}$$

$$\sum_{p \in P:(u,v) \in p} x_p + \sum_{p \in P:(v,u) \in p} x_p \leq 2c_{(u,v)} \; \forall (u,v) \in E, \tag{5}$$

$$\sum_{p \in P:(u,v) \in p} x_p - \sum_{p \in P:(v,u) \in p} x_p = 0 \; \forall (u,v) \in E. \tag{6}$$

This essentially says that the set C consists of all the demands rates $\boldsymbol{\lambda}$, such that they can be allocated to various paths, so that the resulting request rates on each channel are balanced in both directions and respect the capacity constraints of the channel. This set of inequalities is analogous to the capacity region defined in [32].

To get an intuition for the definition of C, let C^∞ denote the set C when all the capacities are infinity. It is easy to see that

$$C^\infty = \left\{ \Lambda : \sum_j \lambda_{ij} = \sum_j \lambda_{ji} \; \forall i \in V \right\}. \tag{7}$$

This just says that the demand rates should be such that in the long run, the total incoming demand to a vertex should be equal to the total outgoing demand or equivalently, we can say that, the demand has to be a circulation [32, Prop. 1].

We will see that C is indeed the capacity region of the payment processing network. First we present the following proposition to show that C is contained in the capacity region.

Proposition 3.4. *Any rate vector $\boldsymbol{\lambda} \notin C$ is not weakly stabilizable by any family of routing algrithms.*

The proof of the proposition is based on strong law of large numbers, and is similar to the argument used in other systems, such as Theorem 4.2.1 in [33], and the details are presented in the technical report [38, Appendix A]. We show that C is indeed the capacity region in Theorem 4.1 in the next section, by presenting a family of routing algorithms that stabilize the payment network for any $\boldsymbol{\lambda} \in C$. However, we will show that this family exhibits a stronger form of stability defined as follows.

Definition 3.5 (Strong Stability). Consider the payment processing network with a given arrival rate vector $\boldsymbol{\lambda}$ and a family of algorithms under which the queue lengths vector $\mathbf{q}$ is a DTMC. This family is said to strongly stabilize the network if for any $\epsilon > 0$, there exists an algorithm in the family that uses at most ϵ rate of on-chain balancing, and the DTMC is positive recurrent.

Note that strong stability implies weak stability because positive recurrence implies existence of a stationary distribution. Therefore, once it is established in Theorem 4.1 that $\boldsymbol{\lambda} \in C$ is strongly stabilizable and so weakly stabilizable, we have that both the notions of stability gives the same capacity region C.

4 THROUGHPUT OPTIMAL ROUTING

In this section, we first present the proposed algorithm and then prove that it is throughput optimal using the Foster-Lyapunov Theorem.

4.1 Routing Algorithm

Before we present the algorithm, we first present one more definition. We define the imbalance, $z_{(u,v)} = q_{(u,v)} - q_{(v,u)}$ as the difference in the outstanding demand from u to v and v to u on each edge (u, v) in the graph. It denotes how much more outstanding requests are from u to v than from v to u. Note that the imbalances can be positive or negative and in particular, $z_{(v,u)} = -z_{(u,v)}$.

The evolution of the DTMC $\mathbf{q}(t)$ is completely characterized by (3), once the routing policy gives the path allocations $\mathbf{x}(t)$ and the amount of on-chain rebalancing $\mathbf{r}(t)$. The proposed family of routing algorithms with parameters M, δ and α is given in Algorithm 1.

Algorithm 1 Routing Algorithm

Parameters: $M > c_{max}, \delta \leq 1, \alpha > 1$
Input: $\mathbf{q}(t), \mathbf{a}(t)$
Initialize: $\mathbf{x}(t) = \mathbf{0}, \mathbf{r}(t) = \mathbf{0}$
# Routing of the demand
for $i \neq j \in V$: $a_{ij}(t) > 0$ **do**
  $p^* = \arg\min_{p \in P_{ij}} \left\{ \sum_{(u,v) \in p} \left(z_{(u,v)}(t) + \frac{\delta}{2\alpha c_{(u,v)}} (q_{(u,v)}(t) + q_{(v,u)}(t)) \right) \right\}$
  $x_{p^*}(t) = a_{ij}(t)$
end for
# On-chain Rebalancing
for $(u, v) \in E$ **do**
  if $q_{(u,v)}(t) > M$ **then**
    $r_{(u,v)}(t) = \begin{cases} 1 & w.p.\ \delta \\ 0 & w.p.\ 1 - \delta \end{cases}$
  end if
end for
Output: $\mathbf{x}(t), \mathbf{r}(t)$

For each source destination pair with an arrival, the algorithm finds a path $p \in P_{ij}$ that has the minimum cost, where the cost of each edge (u, v) is taken to be $z_{(u,v)}(t) + \frac{\delta}{2\alpha c_{(u,v)}}(q_{(u,v)}(t) + q_{(v,u)}(t))$. Thus, this algorithm falls into the class of MaxWeight algorithms, and can be intuitively thought of as follows. Since the

Table 1: Notation

(u, v)	Edge joining node u to node v
P_{ij}	Set of all allowed paths from i to j
$\mathcal{P} = \bigcup_{i,j \in V} P_{ij}$	Set of all the allowed paths
x_p	Amount of demand routed through path p
a_{ij}	Stochastic demand from i to j and it is iid with mean λ_{ij} and variance σ_{ij}^2
$\boldsymbol{\lambda} = (\lambda_{ij})_{i \neq j}$	Vector of demand rates
$c_{(u,v)}$	Capacity of the link $(u, v) \in E$
$q_{(u,v)}(t)$	Outstanding payment requests that need to be served from u to v on the channel (u, v)
$z_{(u,v)}(t)$	Imbalance of the outstanding demand on the edge (u, v), given by $(q_{(u,v)} - q_{(v,u)})$
$s_{(u,v)}(t)$	Amount of demand served on link $(u, v) \in E$ at time t, given by (2)
$y_{(u,v)}(t)$	Total amount of demand routed through the link (u, v) at time t
$r_{(u,v)}(t)$	Amount of on chain rebalancing on link (u, v) at time t

requests on each of our channels are served by pairing them in both the directions, the algorithm tries to equalize the outstanding requests on both sides of the channels by adding more requests to the less loaded side of the channels that have the greatest imbalance. If the capacity of the channels is infinite, all one needs to care about is such balancing of channels because the total arrival rate, then is a circulation as in (7). However, when the capacities are finite, one needs to make sure that the load is distributed appropriately along various paths. So, the algorithm gives more priority to less loaded paths, by adding $\frac{\delta}{2\alpha c_{(u,v)}}(q_{(u,v)}(t) + q_{(v,u)}(t))$ to the weight of each link. Here the coefficient of the weight of the load of the path compared to the imbalance can be tuned using the parameter α. For a given system, an appropriate α can be chosen by conducting simulations on the system. Note that the proposed algorithm picks a single path for each source destination pair. The results can be extended to develop algorithms that pick several paths for privacy reasons, and is a future research direction.

Then the algorithm assigns 1 unit of on chain rebalancing with probability δ for all links (u, v) which are such that $q_{(u,v)}(t) > M$. Note that, the only technical condition we need is the expected on chain rebalancing is δ if the queue length is greater than a certain threshold M which can be realized by performing randomized on chain rebalancing using any distribution with the expected value of the distribution equal to δ. Here, M and δ are parameters for the family of algorithms. Given $\epsilon > 0$, we can tune M and δ so that the on chain rebalancing rate is at most ϵ.

Finally, note that the algorithm can be easily generalized to have different thresholds M for different edges $(u, v) \in E$ with the condition that $M_{(u,v)} \geq c_{(u,v)}$. Although, we consider a single threshold M for the ease of exposition.

4.2 Throughput Optimality

In this subsection, we will prove the main theorem of our paper that the family of algorithms in Algorithm 1 are throughput optimal, and that C is indeed the capacity region of the payment processing network. We now present the main Theorem of the paper.

Theorem 4.1. *Consider the payment processing network with arrival rates $\boldsymbol{\lambda} \in C$, operating under Algorithm 1 with parameters $M > c_{max}$, $1 \geq \delta > 0$ and $\alpha > 1$ such that $\delta < M$. Then the DTMC $\{\mathbf{q}(t) : t \in \mathbb{Z}_+\}$ is positive recurrent with an on-chain rebalancing rate of at most $O(\delta)$. Thus the family of algorithms in Algorithm 1 is strongly stable. Moreover, under Algorithm 1, in steady state, we have*

$$\sum_{(u,v)\in E} \mathbb{E}[q_{(u,v)}] \leq \left(12|E| \sum_{i \neq j \in V} \sigma_{ij}^2 + 2 \sum_{(u,v)\in E} c_{(u,v)}^2\right) \frac{\alpha c_{max}}{\delta^2} + 12|E| \frac{\alpha c_{max}}{\delta} + (4\alpha c_{max}|E|) \frac{M}{\delta} + \frac{c_{max}|E|}{c_{min}} M. \tag{8}$$

Note that the strong stability of the family of algorithms in Algorithm 1 in the theorem follows by making the on-chain rebalancing rate arbitrarily small by choosing small enough δ. Together with Proposition 3.4, the Theorem establishes that C is the capacity region of the payment processing network. Thus the family of algorithms in Algorithm 1 is throughput optimal. Also, note the trade off between the rate of on chain rebalancing used which is $O(\delta)$ and the expected sum of queue lengths with respect to the steady state distribution of the DTMC which is $O(\frac{1}{\delta^2})$. So, with decreasing δ, we reduce the use of on chain rebalancing but increase the time it takes for the transactions to process.

We will prove the theorem using the Foster-Lyapunov Theorem [33] and drift analysis. Before that, we state Lemma 1 which will be useful for the proof.

The following lemma connects the min cost path, the arrival rates with the capacities and the queue lengths at the links, as long as the $\boldsymbol{\lambda}$ is in the capacity region C. In fact Lemma 1 encapsulates all the properties of the capacity region C that we need for the proof of Theorem 4.1.

Lemma 4.2. *If the demand matrix is in the capacity region, i.e. $\boldsymbol{\lambda} \in C$, for any $\mathbf{q}$, we have*

$$\sum_{i \neq j \in V} \lambda_{ij} \min_{p \in P_{ij}} \left\{ \sum_{(u,v)\in p} \left(z_{(u,v)} + \frac{\delta}{2\alpha c_{(u,v)}}(q_{(u,v)} + q_{(v,u)})\right)\right\} \leq \sum_{(u,v)\in E} \frac{\delta q_{(u,v)}}{\alpha}.$$

The reader can refer to the technical report [38, Appendix B] for the details of the proof of Lemma 4.2. We only present the sketch of the proof here.

Sketch of the Proof. The proof is carried out in the following steps:

- We start by defining a linear program with objective function to be zero and the constraints to the constraints of the capacity region.
- Then, as $\boldsymbol{\lambda} \in C$, the primal is feasible and we also argue that the dual is feasible by presenting a feasible point for the dual.

- Then we use weak duality which gives us the dual objective function to be non-negative for all the feasible solutions of the dual LP.
- Then we carefully pick feasible dual variables which gives us the Lemma.

□

After presenting Lemma 4.2 we have the prerequisites to prove Theorem 4.1. The detailed proof is deferred to the technical report [38, Appendix C] due to space constraints. Although we explain the intuition behind the theorem and it's proof below:

Sketch of the Proof of Theorem 4.1. First we show that the DTMC $\{\mathbf{q}(t) : t \in \mathbb{Z}_+\}$ is positive recurrent using Foster-Lyapunov Theorem. This can be done in the following steps:

- First we carefully choose the Lyapunov function as

$$V(\mathbf{q}) \stackrel{\Delta}{=} \sum_{(u,v)\in E} z_{(u,v)}^2 + \sum_{(u,v)\in E} \frac{\delta}{2\alpha c_{(u,v)}} (q_{(u,v)} + q_{(v,u)})^2.$$

- Next, we calculate the one-step drift "$V(q(t+1)) - V(q(t))$" of the defined Lyapunov Function and bound the quadratic terms by known parameters.
- Then, we simplify the expression of the drift using Algorithm 1 and Lemma 4.2.
- Then, we define a finite set of queue lengths and show that the one-step drift is negative outside this set. Thus, by Foster-Lyapunov Theorem, the DTMC is positive recurrent.
- Finally, we argue that the on-chain rebalancing rate is $O(\delta)$ which shows that the Algorithm 1 is Throughput Optimal.

After showing positive recurrence, we use the moment bound theorem [17] to bound the sum of expected queue lengths. This completes the proof. □

After showing the algorithm is throughput optimal which is the main result in the paper, we will provide more intuition for the routing Algorithm 1 by analyzing the fluid model of the system. Although, due to the space constraints, the details are presented in the technical report [38, Section 5].

5 MODIFIED ROUTING ALGORITHM

In this section, we will further discuss Algorithm 1. Even though it is throughput optimal, the simplifications we consider in the model, do not conform with the real Blockchain system. It is desired that the routing algorithm satisfy the Payment Channel Network (PCN) protocol. The conditions being

- Atomicity of the payments, i.e. the payments should be processed immediately or they are rejected.
- The funds in all the edges which are used to process the payment is locked until the whole transaction is processed.

Moreover, it is not desirable to carry out on-chain rebalancing often as it is slow compared to off-chain rebalancing.

We present a modification of the Algorithm 1 which takes into account the PCN protocols and is applicable in real life systems. The algorithm is presented below:

The above algorithm can be understood as follows: Consider the threshold M to be different for each link in the network and set it to the capacity of each link, that is $M = c_{(u,v)}$ for $(u,v) \in E$. Also,

Algorithm 2 Modified Routing Algorithm

Parameters: $M_{(u,v)} = c_{(u,v)}\ \forall (u,v) \in E, \delta \le 1, \alpha > 1$
Input: $\mathbf{q}(t), \mathbf{a}(t)$
Initialize: $\mathbf{x}(t) = \mathbf{0}, \mathbf{r}(t) = \mathbf{0}$
# Routing of the demand
for $i \neq j \in V$: $a_{ij}(t) > 0$ **do**
  $p^* = \arg\min_{p\in P_{ij}} \Big\{ \sum_{(u,v)\in p} \big(z_{(u,v)}(t) + \frac{\delta}{2\alpha c_{(u,v)}}(q_{(u,v)}(t) + q_{(v,u)}(t))\big)\Big\}$
  if $\forall (u,v) \in E : q_{(u,v)}(t) + a_{ij}\mathbb{1}_{(u,v)\in p^*} - s_{(u,v)}(t) \le M_{(u,v)}$ **then**
    $x_{p^*}(t) = a_{ij}(t)$
  else
    # On-chain Rebalancing
    $r_{(i,j)}(t) = \begin{cases} a_{ij}(t) & w.p.\ \delta \\ 0 & w.p.\ 1-\delta \end{cases}$
  end if
end for
Output: $\mathbf{x}(t), \mathbf{r}(t)$

rather than processing \$1 on-chain rebalancing, if the transaction is invoking on-chain rebalancing, then with probability δ process the whole transaction using on-chain rebalancing or reject the transaction otherwise. Now, we have an algorithm which processes payment requests using either only off chain rebalancing or only on chain rebalancing.

Now with this modified algorithm, the metric for the performance will be the fraction of transactions that are processed using on chain rebalancing and the fraction of transactions that are rejected. This metric can be argued to be related to the throughput optimality as follows: the tail probability $\mathbb{P}[q_{(u,v)} \ge c_{(u,v)}]$ for all $(u,v) \in E$ can be bounded using the expected queue length given by (8) which will give us the probability with which a transaction will be either rejected or on chain rebalancing will be invoked. Now, we can tune the parameters δ and α to minimize the appropriate objective. We will verify this in the next section by considering a special case and show that it has good performance in simulations. Also, this modified algorithm conforms with the PCN protocols and it can be argued as follows:

- Under this algorithm, the atomiciy of the payments will be preserved. This can be seen as follows: whenever a path is chosen to route the incoming demand, it is either routed and processed in the same time epoch as $M = c_{(u,v)}$ for all $(u,v) \in E$ or it is rejected with probability $(1-\delta)$ or it is processed using on-chain rebalancing with probability δ. Thus, it preserves the atomicity of the payments.
- Also, note that, it is no longer a single hop system now as either all the single hops are processed in one time epoch as the payment is routed only if the capacity is available or the whole payment is rejected or processed through on chain rebalancing.
- As δ approaches 0, the frequency of on chain rebalancing will be reduced. Thus, on-chain rebalancing won't be carried out in each time epoch. This is desirable as on-chain rebalancing is slower compared to off chain rebalancing.

6 SIMULATIONS

In queueing systems, closed loop policies that use the state information to make scheduling decisions are known to have better performance than open loop policies. Our proposed Algorithm 1 is a closed loop policy since it used the state information, as opposed to Spider [32], which is an open loop policy. In this section, we will compare the performance of Algorithm 1 with that of Spider [32] using simulations.

To study the efficiency of the Algorithm 1, we will drop all the transactions which invokes on chain rebalancing in our algorithm i.e., we will drop any transaction which if added to the payment graph, will lead to at least one edge having weight greater than or equal to M. This will give us the percentage of transactions successful using only off-chain rebalancing. We assume all the weights have equal capacity same as M for simplicity. We use this as a metric (as opposed to calculating the rate of on-chain rebalancing) in order to be consistent with the literature [24, 29, 32].

If we pick $M = c_{(u,v)}$ for every edge in the network and drop all the demand which requires on chain rebalancing, then at any instance, the queue length for each edge is less than its capacity. As the edge has enough capacity to process any buffered transaction at any point of time, this can be thought of as a system with no buffered transactions waiting in the queue to be processed. So although our algorithm is a queuing model, it can be implemented in such a way that there are no transactions waiting in the system. Note that, here we pick M to be different for different edges which is a generalization of the algorithm we presented and the proof of throughput optimality can be easily extended for this system.

6.1 Setup

Simulator: We use Python (networkx package) to create the graph and simulate the transactions. Our simulator finds the path in the graph according to the Algorithm 1 and drops all the transactions which require on-chain rebalacing. Then we add the transaction to the outstanding queues of the edges. This can be thought of as each node has a capacity of M units and if a transaction is processed, it will reduce the capacity of those edges by adding it to the outstanding queue lengths. Also with each transaction that is accepted, $\mathbf{q}$ is updated.

Schemes: We run different variants of our algorithm by varying the parameters involved. We vary M and report the percentage of successful transactions by units. We also report the average imbalance per edge to show the trade off between M and queue length.

Metrics: The metrics of our study are the *average imbalance per edge* and *success ratio*. Success ratio captures the fraction of transactions that were successful.

6.2 Synthetic Data

We consider small graphs on 10 node and 15 edges, and run Algorithm 1, and compare it with an exact exact implementation of Spider network [32] using the exact demand matrix which will give the best possible performance for Spider.

Generation of Transactions: We start by generating a doubly stochastic matrix which is in the capacity region C. Using the generated matrix as the demand rate matrix, we generate a total of 150,000 Poisson distributed transactions. The generated transactions are then used in a random order to simulate the system.

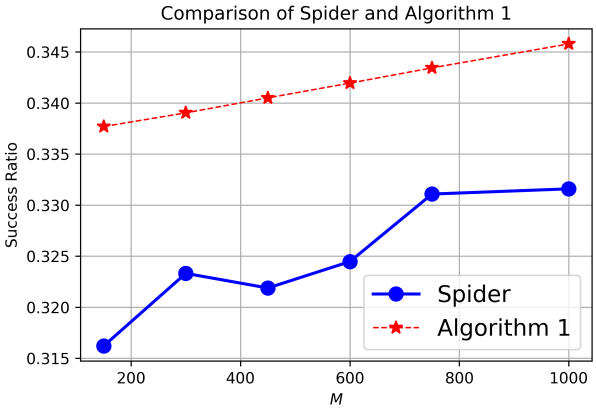

Figure 1: Comparison of Success ratio for Spider and Algorithm 1 for different M on a 10 node graph

Implementation of Spider: To implement the Spider network, throughput is maximized subject to the capacity constraints (fluid LP in [32] is solved) which gives us optimal flow rates through all the paths for routing fluid demand $\boldsymbol{\lambda}$. In the long run, the frequency of usage of the paths is then maintained proportional to the optimal flow rates.

Note that, this corresponds to the exact implementation of the Spider as we are using the exact transaction rate matrix and we are solving the fluid LP before the transactions are processed.

Even if the exact transaction rate matrix is used (which cannot be found exactly in real life implementation), our algorithm consistently performs better than Spider for a wide range of values of M. This is due to the fact that our algorithm actively tries to balance the graph whereas Spider aims to keep the graph balanced in the long run. This can be thought of as an open loop control whereas our algorithm makes decision based on the current state of the system and thus, is a closed loop control.

6.3 Ripple Network

Data set: In this section, we use the data-set provided by Speedy Murmurs [30] which is a sub-graph from the original topology of Ripple. The data-set consists of 59,000 nodes and 191,242 edges. Speedy Murmurs removed the inconsistencies from the original Ripple Data and it was filtered and transformed into a format which was convenient to run simulations on. We are grateful to the authors for providing the data-set. We simulated the system using 50,000 transactions with an average transaction amount of 5279 units, median of 2.94 units and a maximum transaction of $7.16 * 10^7$ units. These are real transactions occurred in the ripple network and provided by Speedy Murmurs.

Approximation of Algorithm 1: Algorithm 1 intends to find a path from a sender to receiver which is the most imbalanced and least loaded path. To implement this algorithm, we need to find the shortest path in a weighted directed graph with negative weights allowed. As this problem is known to be an NP-hard problem (it can be decomposed from Hamiltonian path problem [13]), we use a heuristic implementation of the algorithm on the Ripple data for computational viability.

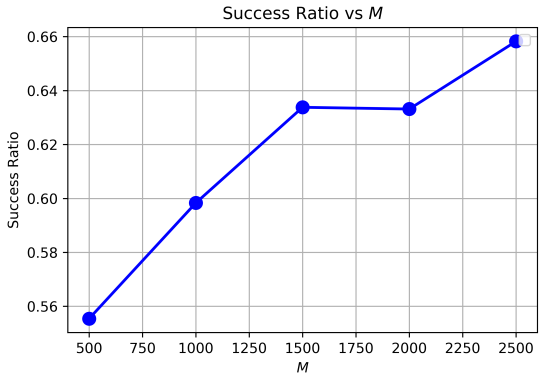

Figure 2: Success ratio of algorithm 1 for the ripple network plotted against the maximum allowable imbalance (capacity: M)

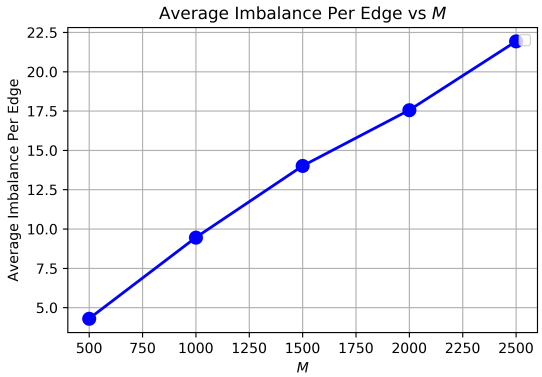

Figure 3: Imbalance per edge in the graph after 50,000 transactions routed through the network using algorithm 1

The heuristic we implement for the real life data is to find k-shortest paths [43] from the sender to receiver by considering only the positive part of the weight $[z_{(u,v)} + \frac{\delta}{2\alpha c_{(u,v)}}(q_{(u,v)} + q_{(v,u)})]^+$ on each edge. Thus, it becomes a problem of finding the shortest path in a weighted directed graph with all the weights non-negative. From those k- shortest path, we will choose the path with minimum actual weight, i.e. the path such that the sum of the weights $z_{(u,v)} + \frac{\delta}{2\alpha c_{(u,v)}}(q_{(u,v)} + q_{(v,u)})$, for all $(u, v) \in p_{ij}$ is minimized. Here, i is the source and j is the destination of the arising transaction. If k is equal to the total possible paths between a sender and receiver, then the heuristic implementation becomes the exact implementation of the algorithm.

Results With the increasing capacity of the edges, the throughput in terms of success ratio increases but stays around 60%. The increasing success ratio with the capacity M is very intuitive as with the increase in capacity of the edges, the system state is allowed to buffer more transactions per edge and thus, more transactions will be processed. Although, the success ratio is not very sensitive to the capacity of the links which can be used to argue that our algorithm does a good job of avoiding imbalance.

The average imbalance per edge increases with the increase in the capacity as with the increasing capacity, we are allowing more buffered transactions which results in the increase of the average imbalance per edge in the steady state. With the increasing success ratio, the success volume also increases which can be seen in Fig. 4. The results are summarized in Fig. 2, Fig. 3 and Fig. 4.

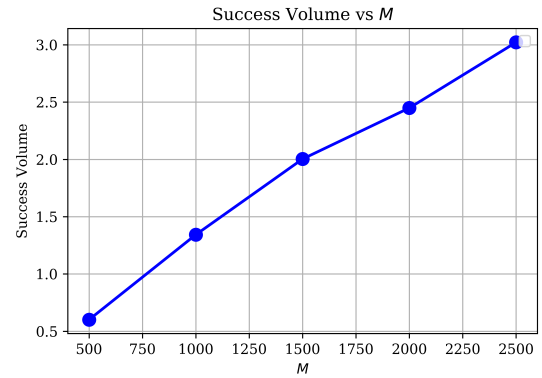

Figure 4: Success volume of algorithm 1 for the ripple network plotted against the maximum allowable imbalance (capacity: M)

7 CONCLUSION AND FUTURE WORK

In this paper, we presented a novel stochastic model to study the routing of payment requests in a Blockchain based payment processing networks. We defined two notions of stability and then characterized the capacity region of the system consisting of all the demand rates under which the system is stable. We then presented a novel MaxWeight like state dependent algorithm and proved that it is throughput optimal. We showed that the rate of on chain rebalancing used is $O(\delta)$ and the expected sum of queue lengths under the stationary distribution of the DTMC is $O(1/\delta^2)$. We argued that a state dependent policy has a better performance than open loop policy by simulating our algorithm and compare it with Spider [32], which uses an open loop policy.

Since the proposed algorithm is NP-hard in general, finding a low-complexity but closed loop algorithm that is throughput optimal is future work. Another future direction is to study throughput maximizing routing algorithms that can be implemented in a distributed manner so that privacy of the transactions is preserved.

REFERENCES

[1] Ivo Adan and Gideon Weiss. 2012. Exact FCFS matching rates for two infinite multitype sequences. *Operations research* 60, 2 (2012), 475–489.
[2] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. 2019. Thickness and information in dynamic matching markets. *Journal of Political Economy* forthcoming (2019).
[3] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. 2017. Efficient dynamic barter exchange. *Operations Research* 65, 6 (2017), 1446–1459.
[4] Frederik Armknecht, Ghassan O Karame, Avikarsha Mandal, Franck Youssef, and Erik Zenner. 2015. Ripple: Overview and outlook. In *International Conference on Trust and Trustworthy Computing*. Springer, https://link.springer.com/chapter/10.1007/978-3-319-22846-4_10, 163–180.
[5] Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. 2017. Pricing and Optimization in Shared Vehicle Systems: An Approximation Framework. In *Proceedings of the 2017 ACM Conference on Economics and Computation (EC '17)*. ACM, New York, NY, USA, 517–517. https://doi.org/10.1145/3033274.3085099
[6] Siddhartha Banerjee, Yash Kanoria, and Pengyu Qian. 2018. State Dependent Control of Closed Queueing Networks with Application to Ride-Hailing. arXiv preprint arXiv:1803.04959.
[7] Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *white paper* 3 (2014), 37.
[8] René Caldentey, Edward H Kaplan, and Gideon Weiss. 2009. FCFS infinite bipartite matching of servers and customers. *Advances in Applied Probability* 41, 3 (2009), 695–730.
[9] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün

Sirer, et al. 2016. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*. Springer, https://link.springer.com/chapter/10.1007/978-3-662-53357-4_8, 106–125.
[10] Pedro Franco. 2014. *Understanding bitcoin*. Wiley Online Library, https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119019138.
[11] Yashar Ganjali, Abtin Keshavarzian, and Devavrat Shah. 2005. Cell switching versus packet switching in input-queued switches. *IEEE/ACM Transactions on Networking (TON)* 13, 4 (2005), 782–789.
[12] Yashar Ganjali, Abtin Keshavarzian, and Devavrat Shah. 2005. Cell switching versus packet switching in input-queued switches. *IEEE/ACM Transactions on Networking (TON)* 13, 4 (2005), 782–789.
[13] Michael R Garey and David S Johnson. 2002. *Computers and intractability (page 213)*. Vol. 29. wh freeman New York, https://pdfs.semanticscholar.org/fa33/c3a89a423f0eb00134fd291894949bf9d54e.pdf.
[14] Arpita Ghosh, Mohammad Mahdian, Daniel M Reeves, David M Pennock, and Ryan Fugger. 2007. Mechanism design on trust networks. In *International Workshop on Web and Internet Economics*. Springer, https://dl.acm.org/citation.cfm?id=1781925, 257–268.
[15] Ye Guo and Chen Liang. 2016. Blockchain application and outlook in the banking industry. *Financial Innovation* 2, 1 (2016), 24.
[16] Itai Gurvich and Amy Ward. 2014. On the dynamic control of matching queues. *Stochastic Systems* 4, 2 (2014), 479–523.
[17] Bruce Hajek. 2006. Notes for ECE 467 Communication Network Analysis.
[18] Maurice Herlihy. 2018. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, https://dl.acm.org/citation.cfm?id=3212736, 245–254.
[19] Yash Kanoria and Pengyu Qian. 2019. Near Optimal Control of a Ride-Hailing Platform via Mirror Backpressure. arXiv:math.OC/1903.02764
[20] Sunny King and Scott Nadal. 2012. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August* 19 (2012).
[21] Siva Theja Maguluri and R Srikant. 2016. Heavy traffic queue length behavior in a switch under the MaxWeight algorithm. *Stochastic Systems* 6, 1 (2016), 211–250.
[22] R Marvin. 2017. Blockchain: The Invisible Technology That's Changing the World. PCMag.
[23] R Mitra. 2017. Lightning protocol & the Raiden network: a beginner's guide. *Retrieved October* 26 (2017), 2018.
[24] Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. 2017. SilentWhispers: Enforcing Security and Privacy in Decentralized Credit Networks. In *24th Network and Distributed System Security Symposium (NDSS 2018)*.
[25] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.
[26] Kim S Nash. 2016. IBM pushes blockchain into the supply chain. The Wall Street Journal. Available online: https://www. wsj. com/articles/ibm-pushes-blockchain-into-the-supplychain-1468528824.
[27] Lam M Nguyen and Alexander L Stolyar. 2018. A queueing system with on-demand servers: local stability of fluid limits. *Queueing Systems* 89, 3-4 (2018), 243–268.
[28] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments.
[29] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. 2018. Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions. In *Proceedings 2018 Network and Distributed System Security Symposium*. NDSS, http://dx.doi.org/10.14722/ndss.2018.23252. https://doi.org/10.14722/ndss.2018.23254
[30] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. 2018. SpeedyMurmurs: Fast and Private Path-based transactions (Software) https://crysp.uwaterloo.ca/software/speedymurmurs/.
[31] Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. 2007. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review* 97, 3 (2007), 828–851.
[32] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathy Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Mohammad Alizadeh, and Giulia Fanti. 2018. High Throughput Cryptocurrency Routing in Payment Channel Networks. arXiv:cs.NI/1809.05088
[33] Rayadurgam Srikant and Lei Ying. 2013. *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, https://www.cambridge.org/us/academic/subjects/engineering/communications-and-signal-processing/communication-networks-optimization-control-and-stochastic-networks-perspective?format=HB.
[34] Bo Tan and Rayadurgam Srikant. 2012. Online advertisement, optimization and stochastic networks. *IEEE Trans. Automat. Control* 57, 11 (2012), 2854–2868.
[35] Leandros Tassiulas and Anthony Ephremides. 1990. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control*. IEEE, https://ieeexplore.ieee.org/document/182479, 2130–2132.
[36] Paul F Tsuchiya. 1988. The landmark hierarchy: a new hierarchy for routing in very large networks. In *ACM SIGCOMM Computer Communication Review*, Vol. 18. ACM, https://dl.acm.org/citation.cfm?id=52329, 35–42. Issue 4.
[37] Richard Lee Twesige. 2015. A simple explanation of Bitcoin and Blockchain technology.
[38] Sushil Mahavir Varma and Siva Theja Maguluri. 2019. Throughput Optimal Routing in Blockchain Based Payment Systems: Link: https://drive.google.com/file/d/1RSY6b-ihXhGvJBdLqMCR_A3Kt3IrZ5lf/view. Available on personal website: https://sites.google.com/view/sushil-varma/home.
[39] Jan Vermeulen. 2017. Bitcoin and Ethereum vs Visa and PayPal–Transactions per second. *My Broadband* 22 (2017).
[40] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151 (2014), 1–32.
[41] David J Wu, Joe Zimmerman, Jérémy Planul, and John C Mitchell. 2016. Privacy-preserving shortest path computation. *arXiv preprint arXiv:1601.02281* (2016).
[42] Lijun Wu, Kun Meng, Shuo Xu, Shuqin Li, Meng Ding, and Yanfeng Suo. 2017. Democratic centralism: A hybrid blockchain architecture and its applications in energy internet. In *2017 IEEE International Conference on Energy Internet (ICEI)*. IEEE, https://ieeexplore.ieee.org/document/7926870, 176–181.
[43] Jin Y Yen. 1971. Finding the k shortest loopless paths in a network. *management Science* 17, 11 (1971), 712–716.