Stable Blockchain Sharding under Adversarial Transaction Generation

Ramesh Adhikari, Costas Busch, Dariusz R. Kowalski

School of Computer and Cyber Sciences

Augusta University

Augusta, GA, USA

SPAA 2024

Nantes, France

June 20, 2024

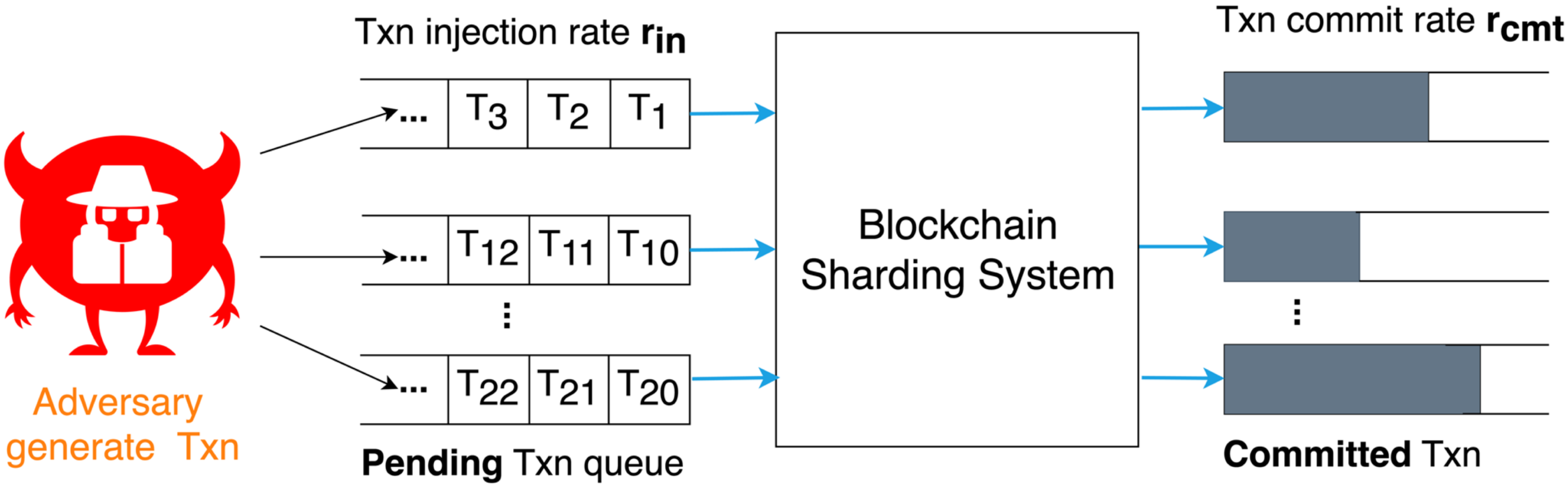
Overview

- Research Purpose
- Background
- Related Works
- Our Proposed Models
- Simulation Results
- Conclusion and Future Work

Research Purpose

Research Purpose

- We aim to make the blockchain sharding system stable under adversarial transaction generation

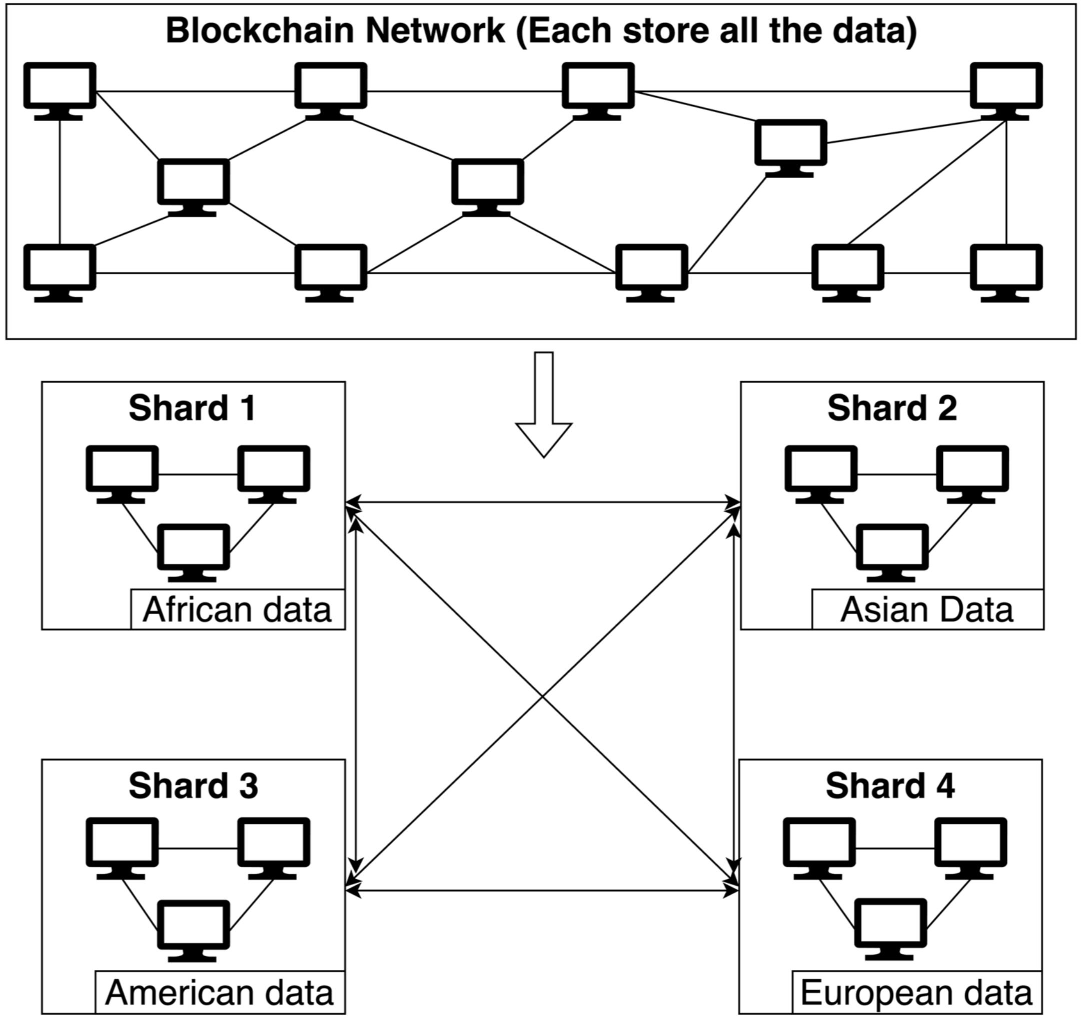


If $r_{cmt} < r_{in}$, then system becomes **unstable**

Background

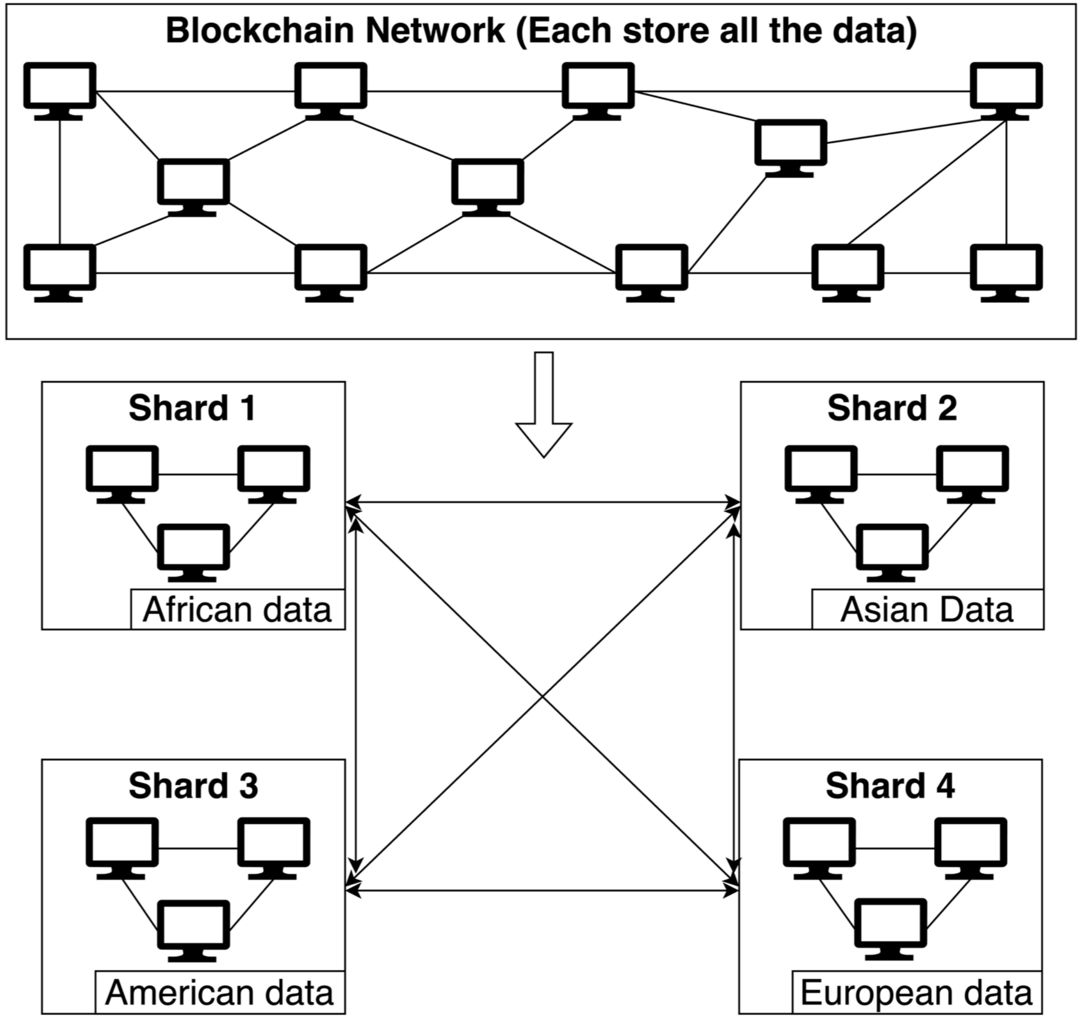
Blockchain Sharding

- Blockchain Sharding for Scalability
  - Divides the blockchain network into smaller groups of nodes called shards



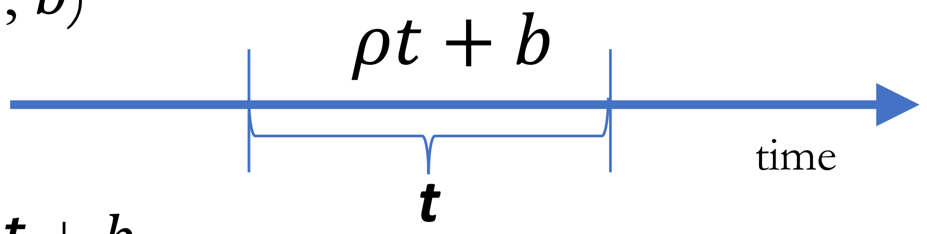
Blockchain Sharding

- Blockchain Sharding for Scalability
  - Divides the blockchain network into smaller groups of nodes called shards
  - Each shard processes transactions in parallel
    - **Commit:** If they access different accounts
    - **Conflict:** If two or more than two transactions try to access same account
    - **Abort:** If condition of transaction is invalid



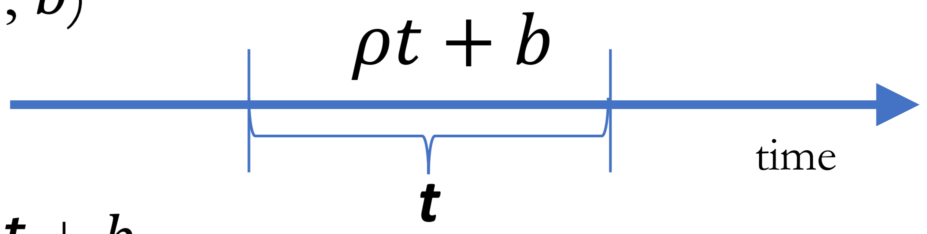
Adversarial Transaction Generation

- Transactions are generated continually by an **adversary** (ρ, b)
  - ρ - injection rate per time unit $(0 < \rho \leq 1)$
  - b – burstiness $(0 < b)$
- Total number of injected transactions in time interval= $\rho \boldsymbol{t} + b$



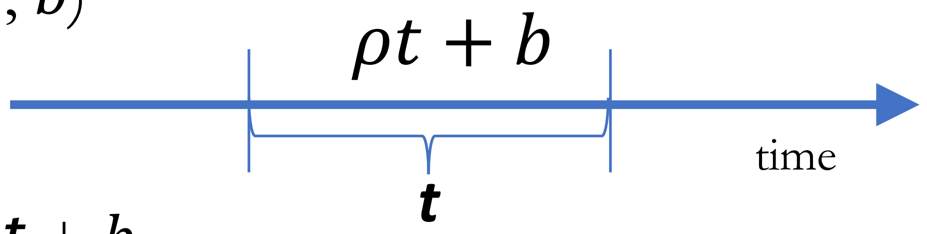
Adversarial Transaction Generation

- Transactions are generated continually by an **adversary** (ρ, b)
  - ρ - injection rate per time unit $(0 < \rho \leq 1)$
  - b – burstiness $(0 < b)$
- Total number of injected transactions in time interval= $\rho \boldsymbol{t} + b$
- Inspired by Adversarial Queueing Theory in networks, where packets are injected continuously



Adversarial Transaction Generation

- Transactions are generated continually by an **adversary** (ρ, b)
  - ρ - injection rate per time unit $(0 < \rho \leq 1)$
  - b – burstiness $(0 < b)$



- Total number of injected transactions in time interval= $\rho \boldsymbol{t} + b$
- Inspired by Adversarial Queueing Theory in networks, where packets are injected continuously
- It models security issues (DOS attack) in blockchain sharding system:
  - The adversary attempts to make the system unstable by injecting too many transactions

Contributions

- **Upper bound:** Provide absolute upper bound on the maximum transaction injection rate $\rho \leq max\left\{\frac{2}{k+1}, \frac{2}{\sqrt{2s}}\right\}$ for which scheduler is feasible

s: total number of shards
k: max. no. of shards accessed by each transaction
c: some positive constant
d: worst distance of any transaction to the shards it access

Contributions

- **Upper bound:** Provide absolute upper bound on the maximum transaction injection rate $\rho \le max\left\{\frac{2}{k+1}, \frac{2}{\sqrt{2s}}\right\}$ for which scheduler is feasible
- **Basic scheduler:** Stable scheduling algorithm with upper bound on transaction injection rate: $\rho \le max\left\{\frac{1}{18k}, \frac{1}{18\sqrt{s}}\right\}$

s: total number of shards
k: max. no. of shards accessed by each transaction
c: some positive constant
d: worst distance of any transaction to the shards it access

Contributions

- **Upper bound:** Provide absolute upper bound on the maximum transaction injection rate $\rho \leq max\left\{\frac{2}{k+1}, \frac{2}{\sqrt{2s}}\right\}$ for which scheduler is feasible
- **Basic scheduler:** Stable scheduling algorithm with upper bound on transaction injection rate: $\rho \leq max\left\{\frac{1}{18k}, \frac{1}{18\sqrt{s}}\right\}$
- **Distributed scheduler:** Stable scheduling algorithm with upper bound on transaction injection rate: $\rho \leq \frac{1}{cd\log^2 s} max\left\{\frac{1}{k}, \frac{1}{\sqrt{s}}\right\}$

s: total number of shards
k: max. no. of shards accessed by each transaction
c: some positive constant
d: worst distance of any transaction to the shards it access

Contributions

- **Upper bound:** Provide absolute upper bound on the maximum transaction injection rate $\rho \leq max\left\{\frac{2}{k+1}, \frac{2}{\sqrt{2s}}\right\}$ for which scheduler is feasible
- **Basic scheduler:** Stable scheduling algorithm with upper bound on transaction injection rate: $\rho \leq max\left\{\frac{1}{18k}, \frac{1}{18\sqrt{s}}\right\}$
- **Distributed scheduler:** Stable scheduling algorithm with upper bound on transaction injection rate: $\rho \leq \frac{1}{cd \log^2 s} max\left\{\frac{1}{k}, \frac{1}{\sqrt{s}}\right\}$
- **Simulation results:** Provide simulation results of proposed scheduler

s: total number of shards
k: max. no. of shards accessed by each transaction
c: some positive constant
d: worst distance of any transaction to the shards it access

Related Works

Related Works

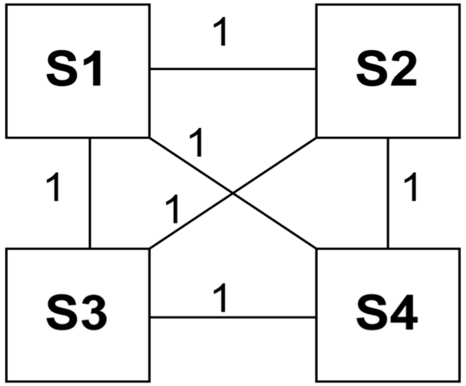
- No prior study in the stability analysis of blockchain sharding systems
- Prior works in Transactional Memory [1] but cannot directly apply to blockchain sharding
- Related works in adversarial queuing[2] provide stability analysis in routing and broadcasting algorithms

[1] Costas Busch, Bogdan S Chlebus, Dariusz R Kowalski, and Pavan Poudel. Stable Scheduling in Transactional Memory. In Algorithms and Complexity: CIAC 2023, 2023
[2] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P Williamson. Adversarial queuing theory. Journal of the ACM (JACM), 2001

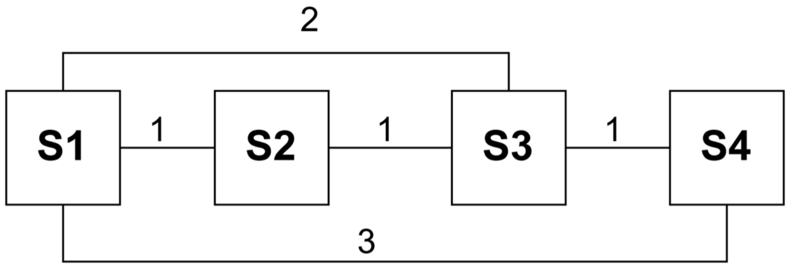
Our Proposed Models

Proposed Models (Schedulers)

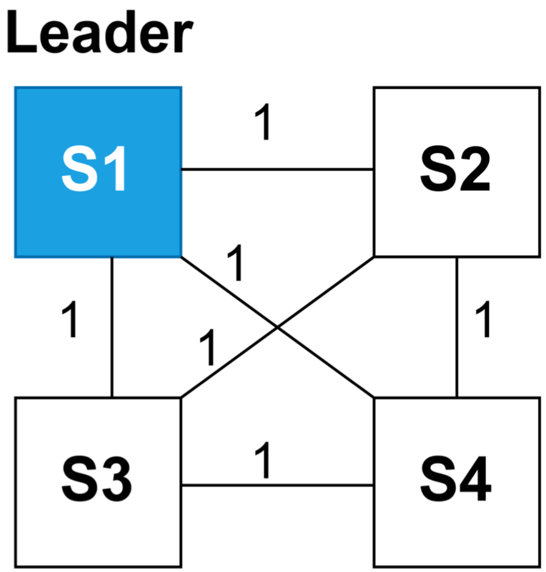
- **Basic Scheduler**
  - Shards are in **uniform communication** model**:**
    - Any two shards are at a unit distance away



- **Distributed Scheduler**
  - Shards are in **non-uniform communication** model:
    - Distance between any two shards ranges from 1 to *d*, where *d* is diameter of shard graph

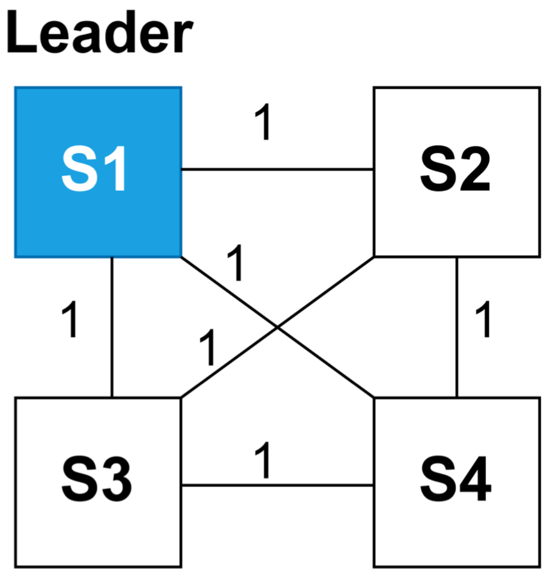


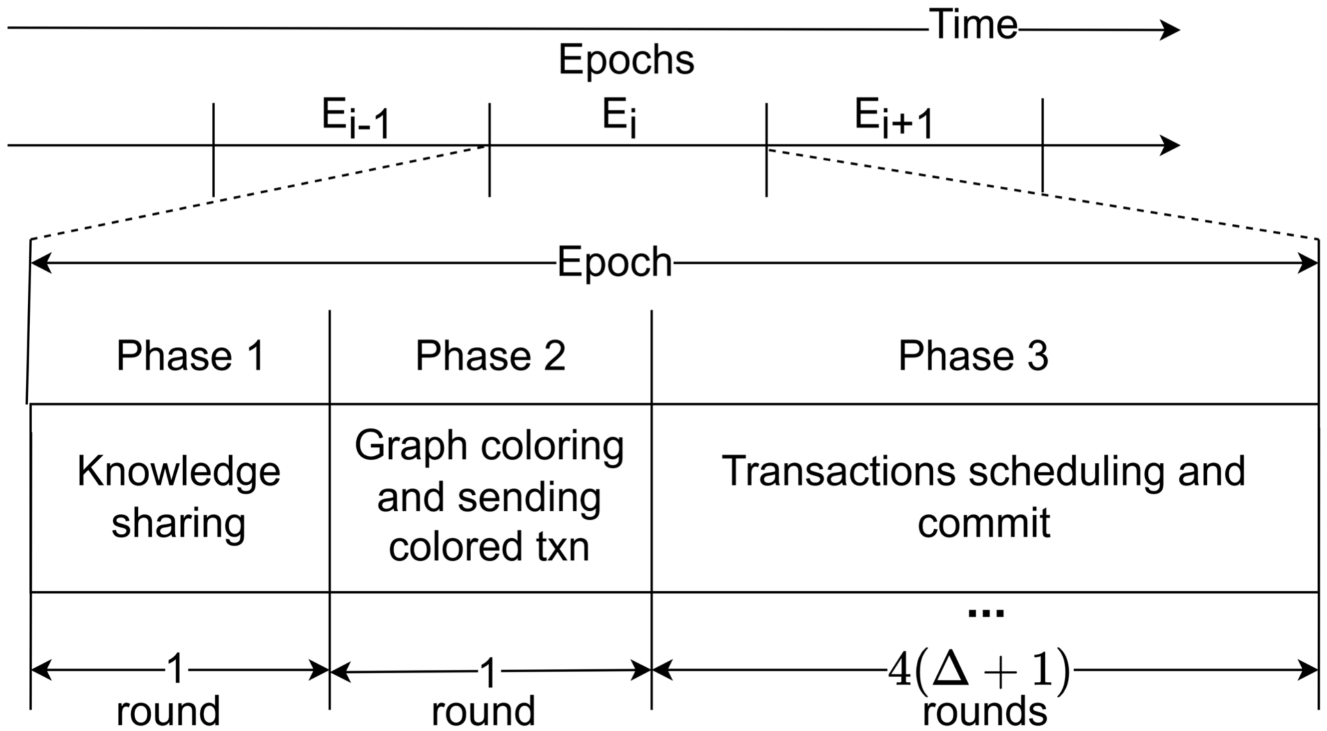
Basic Scheduler



- One of the shard is a leader, which determines schedule of transactions by coloring

Basic Scheduler

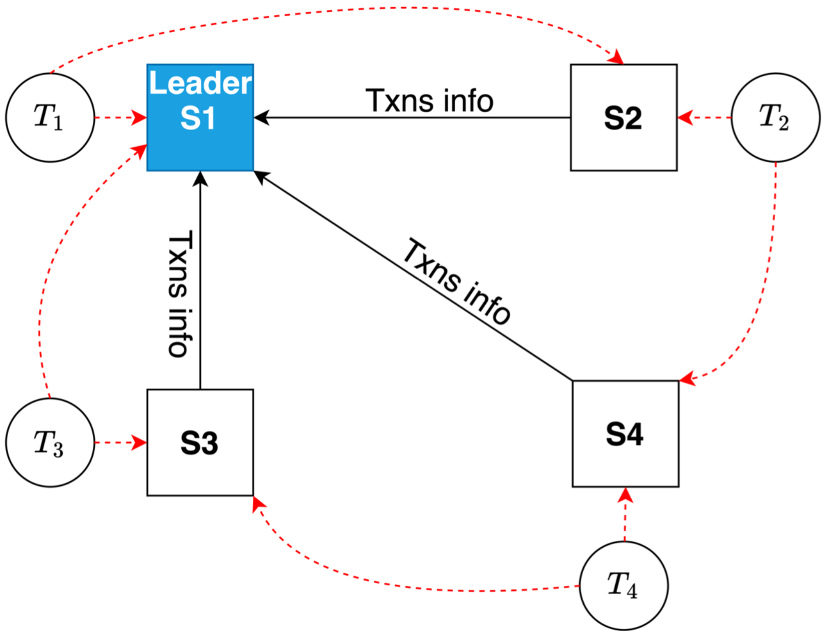


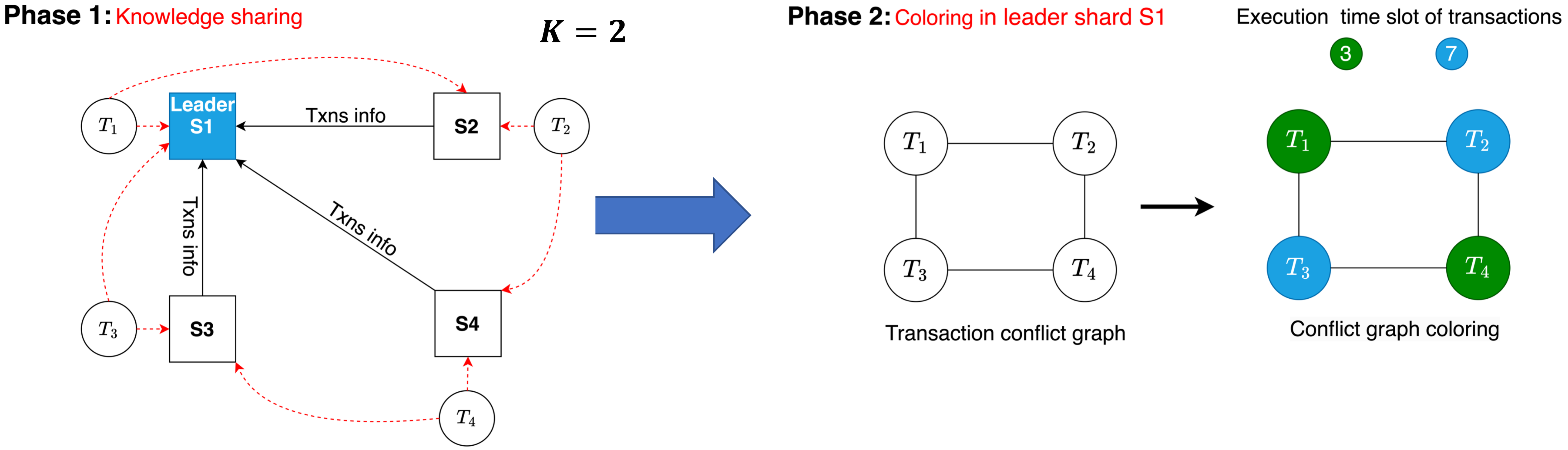


- One of the shard is a leader, which determines schedule of transactions by coloring
- (ρ, b)-adversary and Algorithm run in epochs

Phase 1: Knowledge sharing

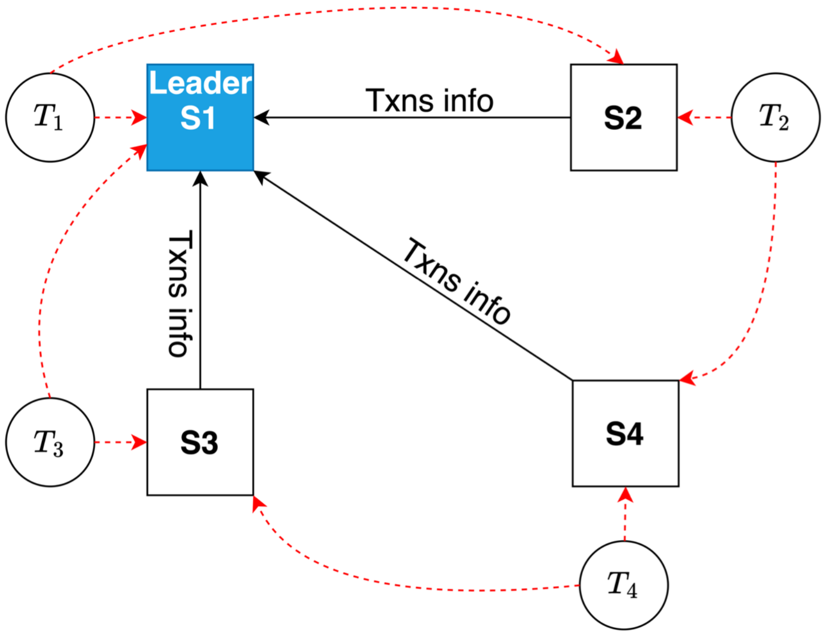
$K = 2$




Phase 1: Knowledge sharing
$K = 2$
Leader S1
S2
S3
S4
Txns info
Txns info
Txns info
T_1
T_2
T_3
T_4
Phase 2: Coloring in leader shard S1
Execution time slot of transactions
3
7
T_1
T_2
T_3
T_4
Transaction conflict graph
T_1
T_2
T_3
T_4
Conflict graph coloring

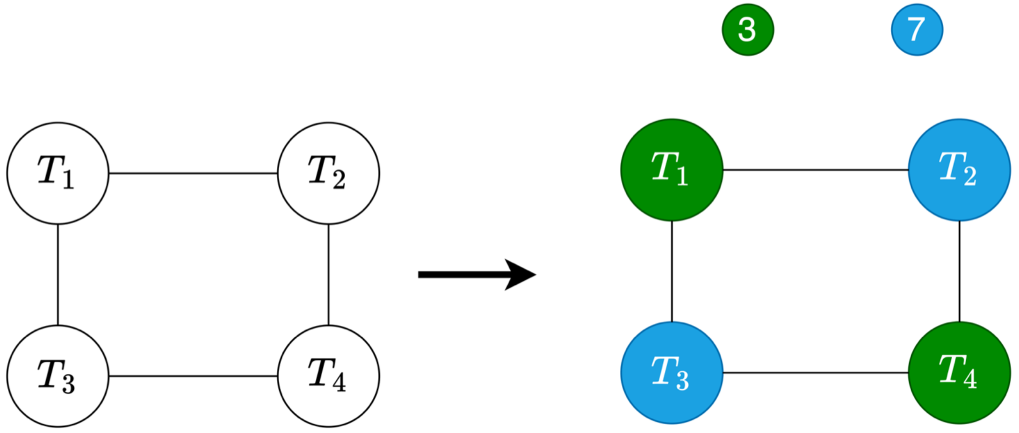
Phase 1: Knowledge sharing

$K = 2$



Phase 2: Coloring in leader shard S1

Execution time slot of transactions

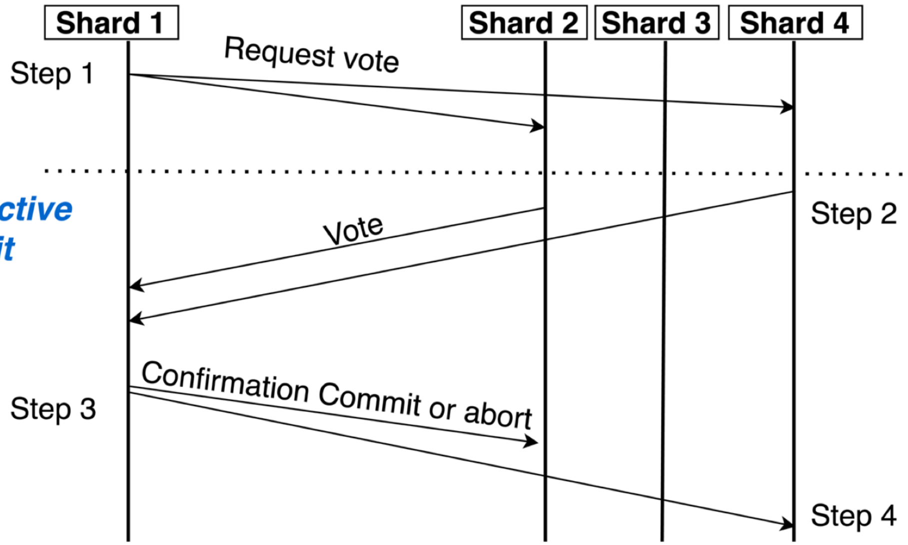


Transaction conflict graph

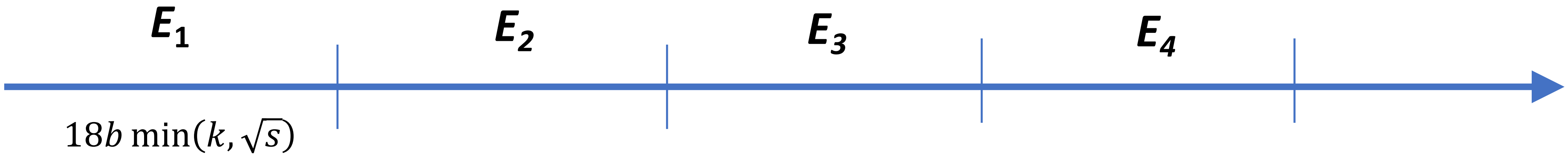
Conflict graph coloring

Phase 3: confirmation and commit

Split Txn in subTxn and send to respective shard for confirmation and commit

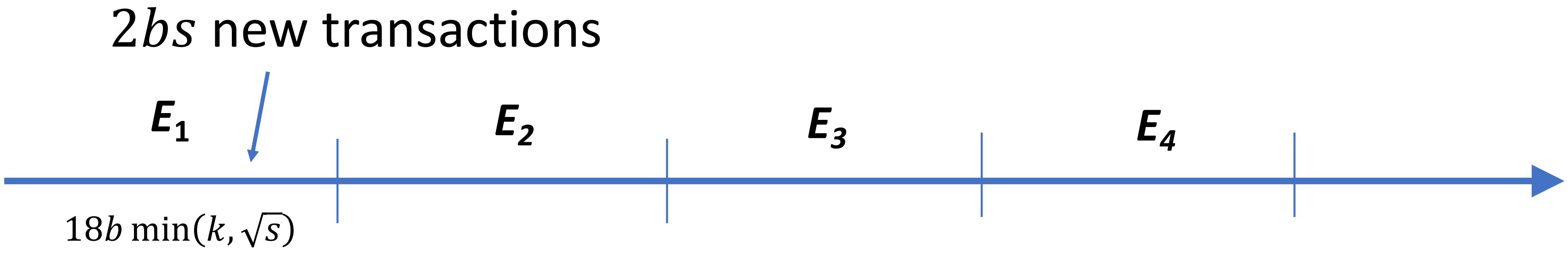


Basic Scheduler



Basic Scheduler

For injection rate $\rho \leq max\left\{\frac{1}{18k}, \frac{1}{18\sqrt{s}}\right\}$ at most $2bs$ new transactions per epoch



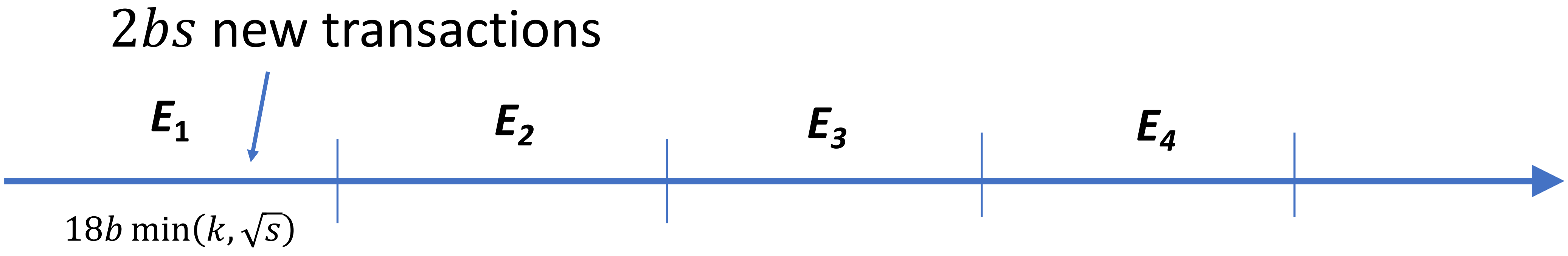
Basic Scheduler

For injection rate $\rho \leq max\left\{\frac{1}{18k}, \frac{1}{18\sqrt{s}}\right\}$ at most $2bs$ new transactions per epoch

Number of transactions per shard:

$$\rho \cdot Epoch\ length + b \leq max\left\{\frac{1}{18k}, \frac{1}{18\sqrt{s}}\right\} \cdot 18b \min(k, \sqrt{s}) + b \leq b + b = 2b$$

For all s shards: $2bs$



Basic Scheduler

- In the set of 2bs pending transactions in epoch, a transaction conflicts with at most $2bk$ other transactions
  - Each transaction accesses k shards and each shard is accessed by b transactions

Basic Scheduler

- In the set of 2bs pending transactions in epoch, a transaction conflicts with at most $2bk$ other transactions
  - Each transaction accesses k shards and each shard is accessed by b transactions
- There are at most $2bk + 1$ conflict-free sets of transactions (greedy coloring)

Basic Scheduler

- In the set of 2bs pending transactions in epoch, a transaction conflicts with at most $2bk$ other transactions
  - Each transaction accesses k shards and each shard is accessed by b transactions
- There are at most $2bk + 1$ conflict-free sets of transactions (greedy coloring)
- Knowledge sharing and coloring takes 1 round each and each color takes 4 rounds for the confirmation and commits
- So total required rounds = 1+1+4(2bk+1) = 8bk+6

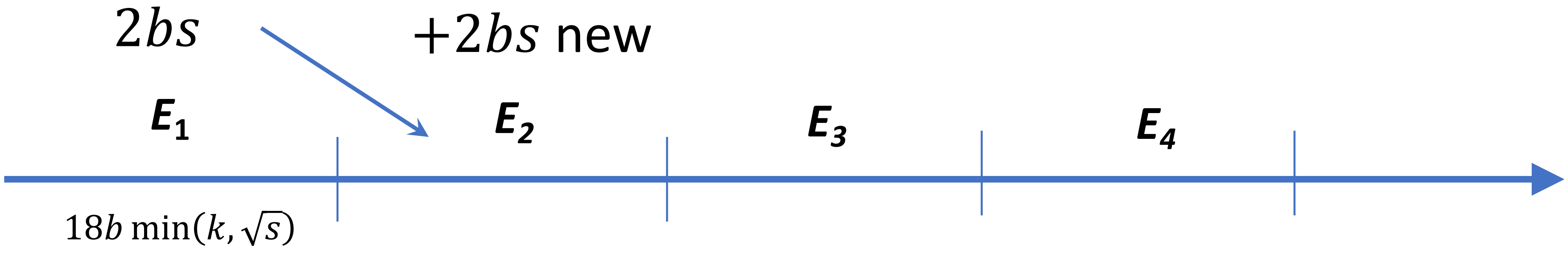
Basic Scheduler

- In the set of 2bs pending transactions in epoch, a transaction conflicts with at most $2bk$ other transactions
  - Each transaction accesses k shards and each shard is accessed by b transactions
- There are at most $2bk + 1$ conflict-free sets of transactions (greedy coloring)
- Knowledge sharing and coloring each takes 1 round and each color takes 4 rounds for the confirmation and commits
- So total required rounds = 1+1+4(2bk+1) = 8bk+6

We consider epoch length is at most 18 $b\ k > 8bk + 6$ which is sufficient to process all transactions

Basic Scheduler

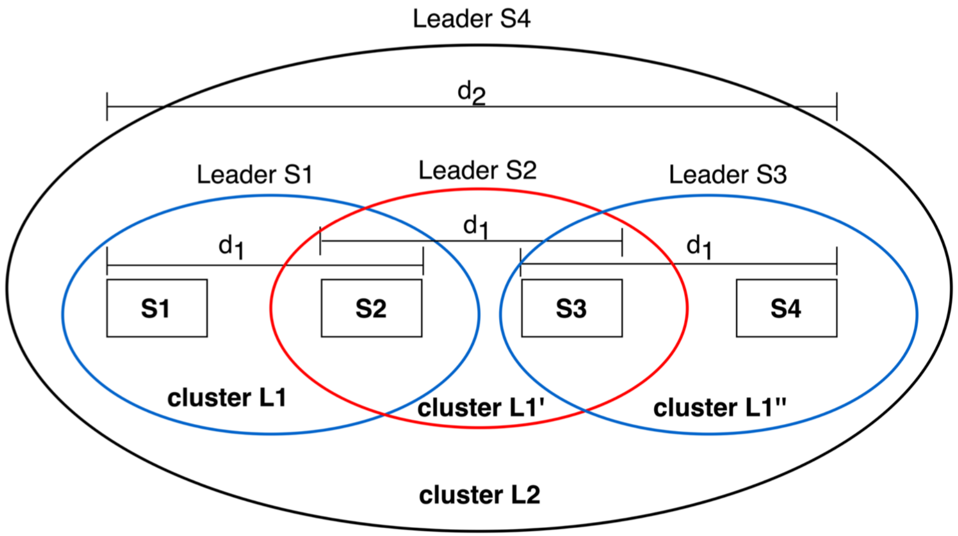
Transactions generated in an epoch execute by the end of next epoch



Hence, at most $4bs$ pending transactions per epoch

Stability!

Distributed Scheduler



- Algorithm uses hierarchical clustering of the shards
- Each cluster has its own leader which colors and determines the schedule of transactions of that cluster
- Each transaction finds their cluster according to the distance between transaction and accessing shards
- Similar approach as in Basic scheduler in each cluster. However, conflict-free sets are discovered in a distributed manner

Distributed Scheduler

- Intervals $I_1, I_2, \ldots$each has length $cbd \log^2 s \min(k, \sqrt{s})$ rounds
  - Where c is some positive constant c and d is the maximum distance between transaction and it accessing shards.

$$\boxed{\text{Stable for injection rate: } \rho \le \frac{1}{cd \log^2 s} max\left\{\frac{1}{k}, \frac{1}{\sqrt{s}}\right\}}$$

We show:

$$\boxed{\text{For injection rate } \rho \le \frac{1}{cd \log^2 s} max\left\{\frac{1}{k}, \frac{1}{\sqrt{s}}\right\} \text{ at most } 4bs \text{ pending transactions per interval}}$$

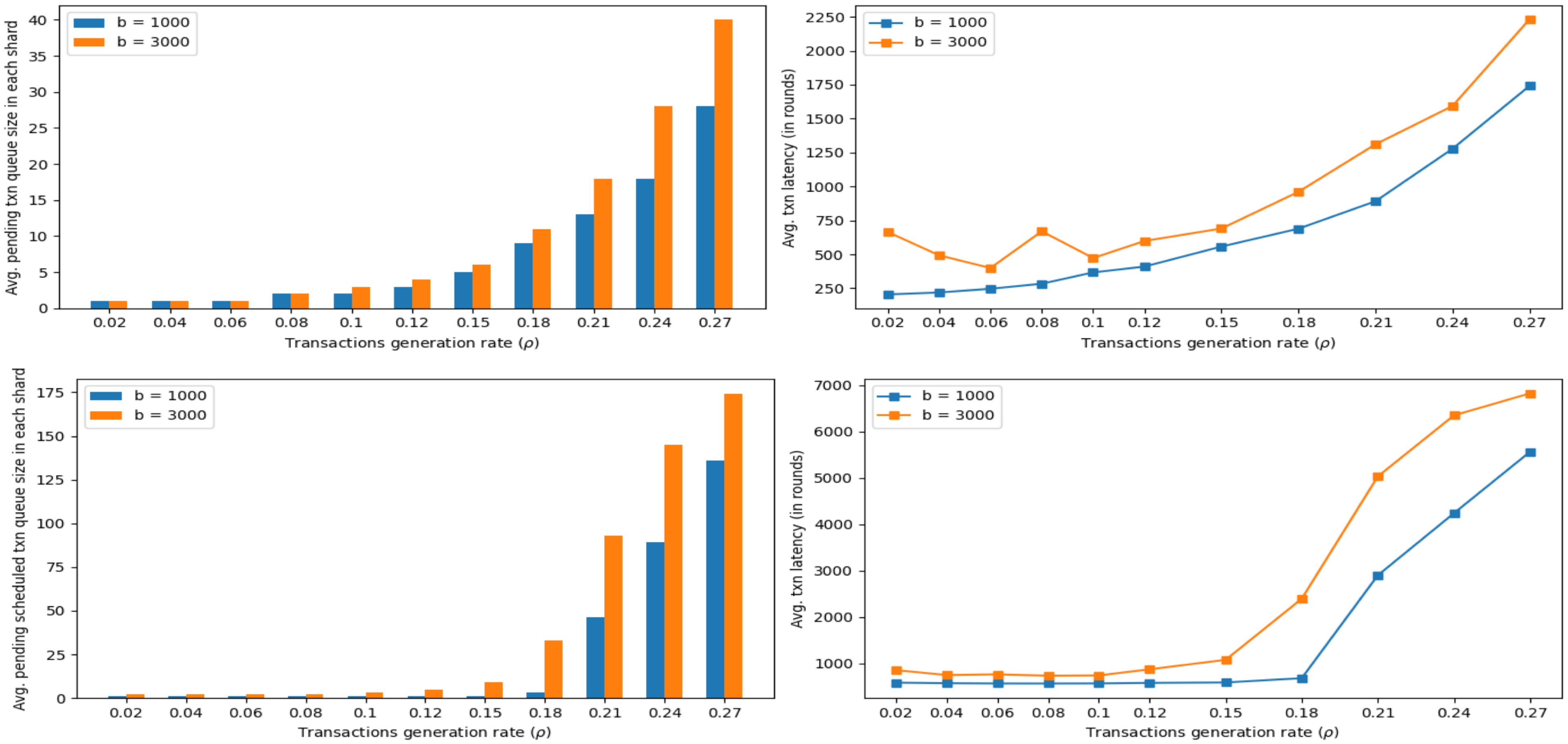
Stability!

Simulation Results

Simulation Results

- Simulated using python, in MacBook Pro 10-core, 32 GB RAM
- Total number of shards (s)=64
- Number of accounts = 64 (one account per shard)
- Max. no. of shards accessed by each transaction (k) = 8

Simulation Results



Basic Scheduler

Distributed Scheduler

On the left, the average pending transactions in the queue of each shard is shown versus ρ. On the right, the average transaction latency measured in rounds is plotted against ρ.

Conclusion and Future Work

Conclusions and future work

- Studied blockchain sharding system with adversarial transaction generation:
  - Designed **Basic Scheduler** and **Distributed Scheduler** and provide upper bound on transaction injection rate

- Implication
  - Blockchain designers can use our results to design stable blockchain system, which is resilient to Denial of Service Attack (DOS)

- Future work**:**
  - Exploring efficient communication mechanism between shards with minimum message size
  - Exploring the systems with heavy traffic

Thank You!

Questions?