Transaction Confirmation in Proof-of-Work Blockchains: Auctions, Delays and Droppings

PRIN Nirvana kickoff meeting

Simonetta Balsamo, Andrea Marin, Sabina Rossi, Isi Mitrani, Ivan Malakhov

Università Ca'Foscari Venezia, IT
University of Newcastle, UK

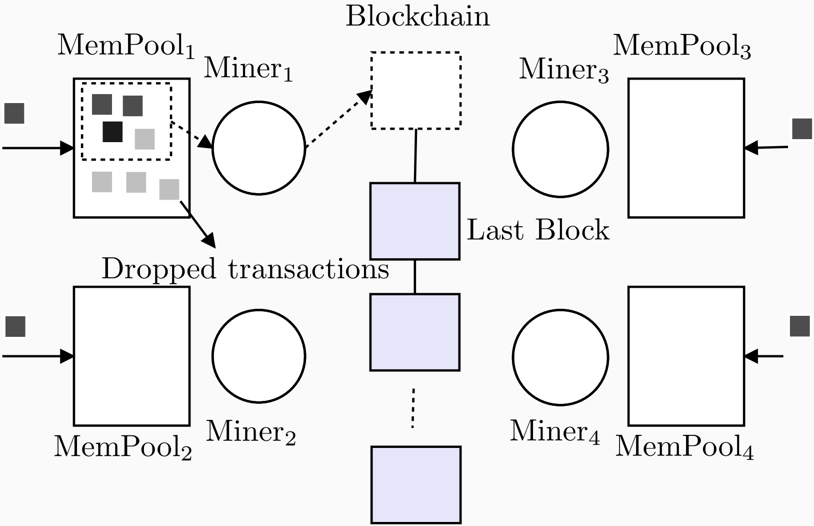
Talk overview

Introduction

Blockchain review

- Distributed ledgers for permanent and unmodifiable storage of data
- Usually associated with cryptocurrencies (BitCoin, Ethereum, ...)
- Several ways for guaranteeing data integrity
  - In this paper we consider the most common Proof of Work (PoW) system
- Data are encoded in transactions and transactions are grouped in blocks
- Once a block is consolidated, it cannot be changed and the transactions contained in it are confirmed

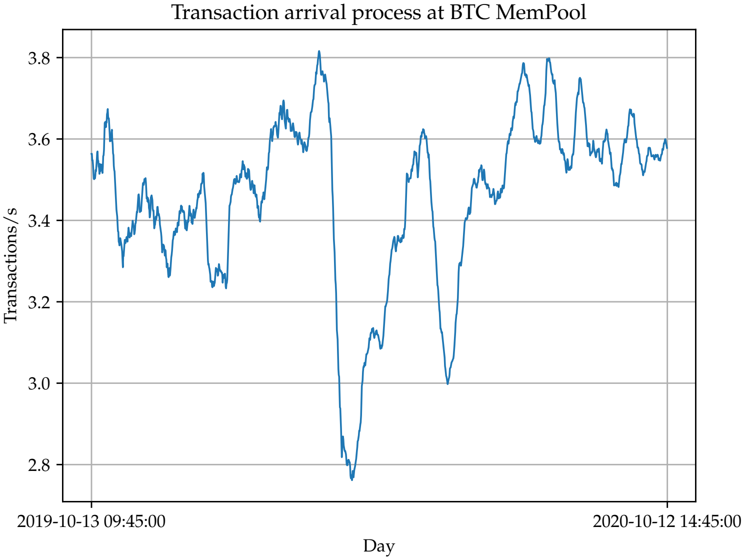
Sketch of the PoW mechanism



How are transactions chosen from the MemPool?

- Each transaction offers a fee to pay the work of the miners
- The protocol does not establish an order of service for the transactions
- In general, miners choose to insert in the block the transactions with the highest fee per Byte to maximize their profit
  - Notice that blocks have a maximum size
  - In general, blocks can be generated at a certain maximum average speed (e.g., 1 block every 10 minutes in Bitcoin)
  - This imposes the maximum throughput of the system!

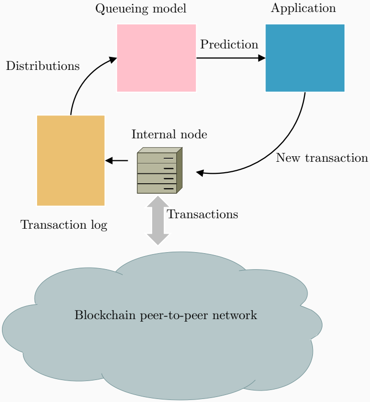
Intensity of the arrival process in BTC blockchain



Problem statement

- Given the operating conditions of the blockchain, is it possible to determine the expected transaction consolidation time given the offered fee?

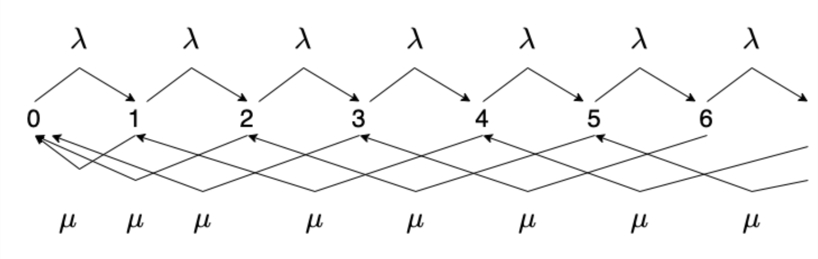
Application



Two perspectives

- System-centric: how long, on average, does it take to confirm a transaction with a certain fee? (stationary analysis, neglect the current Mempool size)
- User-centric: as a user, what should be my fee to satisfy a certain requiremenet of the expected confirmation time? (transient analysis, Mempool size matters!)

A common mathematical framework

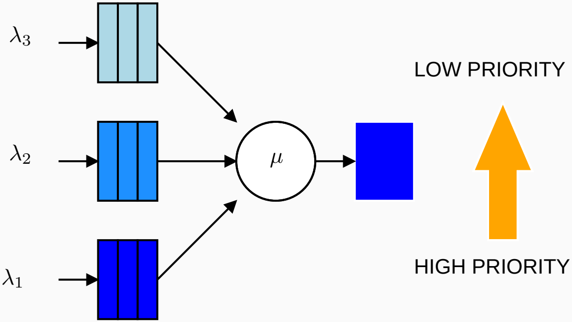


- B Block size (in figure $B = 3$), in BTC $B \simeq 2400$
- μ: block generation rate, in BTC $\mu = 1/600$s
- λ: arrival rate, in BTC below 4 transactions/s

The system-centric model

Modeling assumptions and notations

- We consider a multi-class $M/M^B/1$ queueing system
  - Poisson arrival, exponential service time, batches of size B
- Each class has a strong priority
- Batches are filled with jobs from high to low priority
  - Once a job is in the batch, it may be removed if a job with higher priority arrives



Stability and reneging

- Class i is stable if and only if $\Lambda_i < \mu B$, where $\Lambda_i = \sum_{j=1}^{i} \lambda_j$
- For unstable classes we study two types of reneging policies:
  - After an exponential random time with rate γ_i class i transaction are dropped
  - The MemPool has finite capacity
- Henceforth, we assume that only the lowest priority class may be unstable

Stationary solution of the highest priority class

- Let z_1 be the unique real root of the polynomial

$$\lambda_1(1-z) - \mu z(1-z^B)$$

  such that $0 < z_1 < 1$
- $p_{1,n} = (1-z_1)z_1^n$
- The expected Mempool occupancy is $L_1 = z_1/(1-z_1)$
- The expected Confirmation time is obtained by Little law: $C_1 = L_1/\lambda_1$

Stationary solution of the lower priority classes without reneging

- Consider class $i > 1$ and assume you know L_i for all classes lower than i
- Let z_i be the unique real root of the polynomial

$$(\lambda_1 + \lambda_2 + \cdots \lambda_i)(1 - z) - \mu z(1 - z^B)$$

  such that $0 < z_i < 1$
- The expected Mempool occupancy of jobs of classes $1, \ldots, i$ is $L_i^c = z_i/(1 - z_i)$
- Therefore $L_i = L_i^c - \sum_{j=1}^{i-1} L_j$
- The expected Confirmation time is obtained by Little law: $C_i = L_i/\lambda_i$

Approximate solution of the lowest priority class

- The expected space in a batch seen by the lowest priority class is:

$$b = \sum_{n=0}^{B-1} (B-n)p_n$$

- We approximate the batch service with single service with rate μb
  - The approximation is good in heavy-load which is the lowest priority class regime
- We give a recursive scheme to approximate the probability of dropping, expected response time and average MemPool size in the case of timeout reneging
- The case of bounded queue is handled as a $M/M/1/K$ system

Numerical validation: Methodology

- We validate our model on the BTC blockchain
- We collect data from blocks in time intervals of 7 hours
- For each transaction we check the *first seen* field in `blockchain.com` and `bitaps.com`
- We cluster the transactions into 5 classes based on their fee per Byte offered
- We measure the expected consolidation time for each class
- The intensity of the arrival process per class is obtained by monitoring the MemPool in a BTC node in the reference time interval.

Classes and distribution

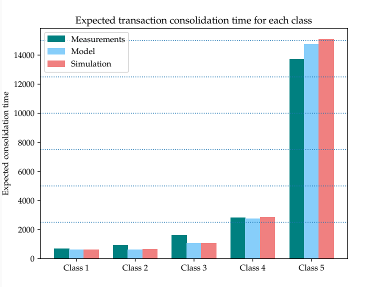
Class	Range [S/B]	Dist. in heavy load	Dist. in moderate-load
1	$[100, \infty)$	0.069	0.066
2	$[60, 100)$	0.235	0.216
3	$[40, 60)$	0.315	0.152
4	$[20, 40)$	0.184	0.096
5	$[0, 20)$	0.196	0.470

S/B: Satoshi per Byte

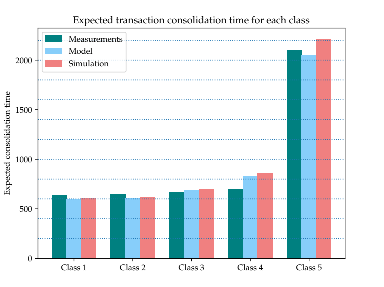
Notice that on high-load the average of the distribution becomes higher

Expected consolidation time

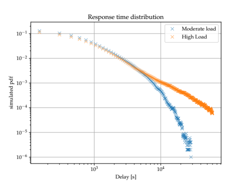
Heavy load



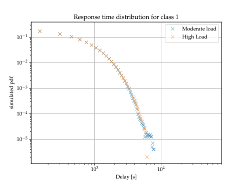
Moderate load



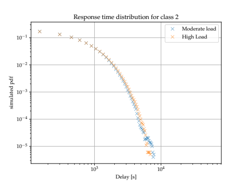
Distribution of the response time (simulated)



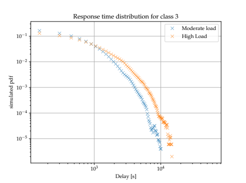
(a) Overall simulated response time distribution .



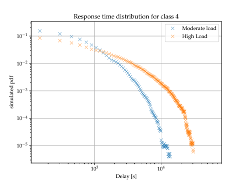
(b) Simulated response time distribution for class 1.



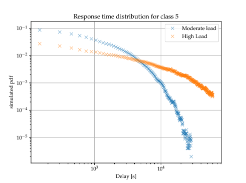
(c) Simulated response time distribution for class 2.



(d) Simulated response time distribution for class 3.

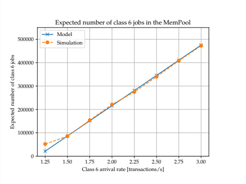


(e) Simulated response time distribution for class 4.

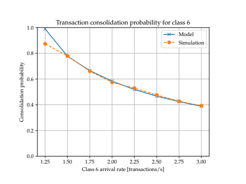


(f) Simulated response time distribution for class 5.

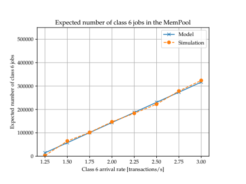
Analysis of the reneging



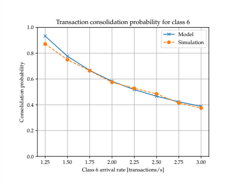
(a) Expected number of class 6 transactions in the MemPool. The expected reneging time is 72 hours.



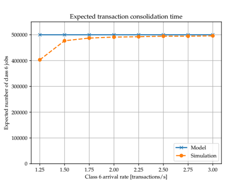
(b) Consolidation probability for class 6 transactions. The expected reneging time is 72 hours.



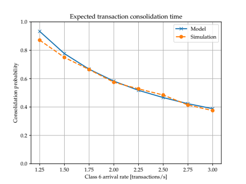
(c) Expected number of class 6 transactions in the MemPool. The expected reneging time is 48 hours.



(d) Consolidation probability for class 6 transactions. The expected reneging time is 48 hours.



(e) Expected number of class 6 transactions in the MemPool with buffer capacity for this class is $5 \cdot 10^5$.



(f) Consolidation probability for class 6 transactions. The buffer capacity for this class is $5 \cdot 10^5$.

Final observations on the system-centric model

- We have presented a queueing model for the analysis of the expected consolidation time of transactions in blockchain based on Proof-of-Work
    - The assumption is that miners try to maximise their profit by selecting the transaction with the highest fee per byte
- The validation of the model with BTC data showed a good accuracy
- We studied the efficiency of two types of reneging in the case of overloaded systems

The user-centric model

Observation

- When a user wants to send a transaction he/she is aware of:
  - The arrival intensity of transaction
  - The distribution of the fees offered up to that point
  - The population and the distribution of the fees in the Mempool
- If the user offers f, he/she sees a system populated only by the jobs with a cost greater than f and all the jobs with fee grater than f will overtake it
- His/her transaction will be served when all more expensive transactions have been served, i.e., when the *filtered* Mempool is empty

Roadmap

- Assume you have cheapest transaction
- Discretization of the model time: at each block generation, you have a tic
- The number of arrival between two tics is geometrically distributed: $a_j = \beta\alpha^j$ with $\beta = 1 - \alpha$
- Compute the expected number of jumps to the absorption in state 0 given the initial state
    - This is done by resorting to the generating function methods
    - Some cumbersome mathematical details are present in the proof

Main result

Let M_1^Y be the expected number of steps to reach the absorbing state when the queue satisfies the stability condition starting from state Y. Then, the following recursive scheme can be used to derive M_1^Y:

$$\begin{cases} M_1^1 = P_1'(1) \\ M_1^{Y+1} = M_1^Y + \frac{T_{Y-1}}{\alpha^{Y-1}}\left(M_1^1 + \frac{\beta}{\alpha}\right) - \frac{T_Y}{\alpha^Y} M_1^1 \,, \end{cases} \tag{1}$$

where:

$$T_Y \triangleq \sum_{c=0}^{\lfloor \frac{Y}{B+1} \rfloor} (-1)^{c+1} \binom{Y - Bc}{c} \alpha^{Bc} \beta^c \,. \tag{2}$$

Main result: getting $P_1'(1)$

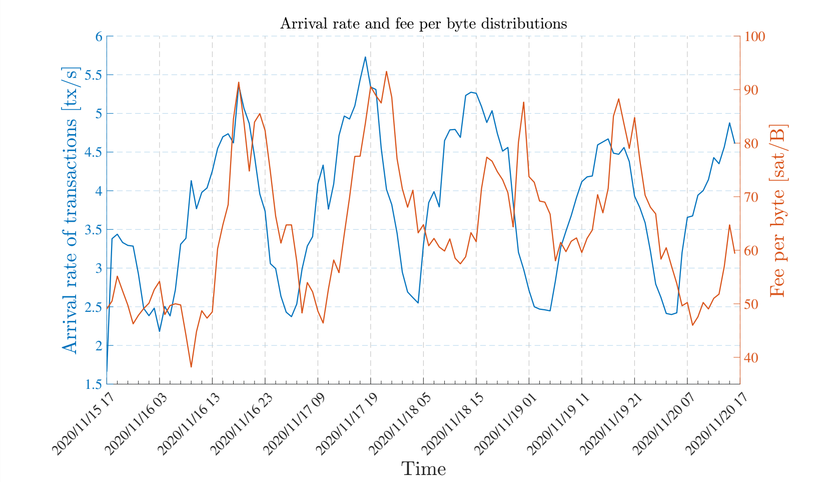
We prove that:

$$P_1'(1) = \left(\frac{z}{1-z}\right)\frac{1-\alpha}{\alpha}\frac{1}{\mu}\,.$$

and z is the unique root in $(0,1)$ of the polynomial

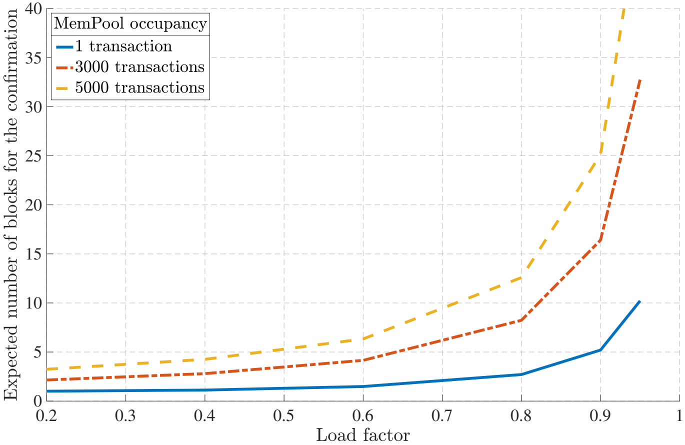
$$\mu x^{B+1} - (\lambda+\mu)x + \lambda$$

State of the art: smartfeeprediction



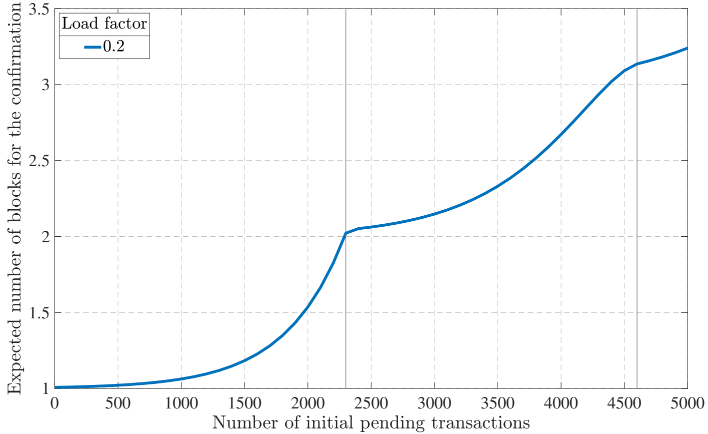
The sytem is reactive! (The queueing model is proactive)

The impact of the initial Mempool occupancy



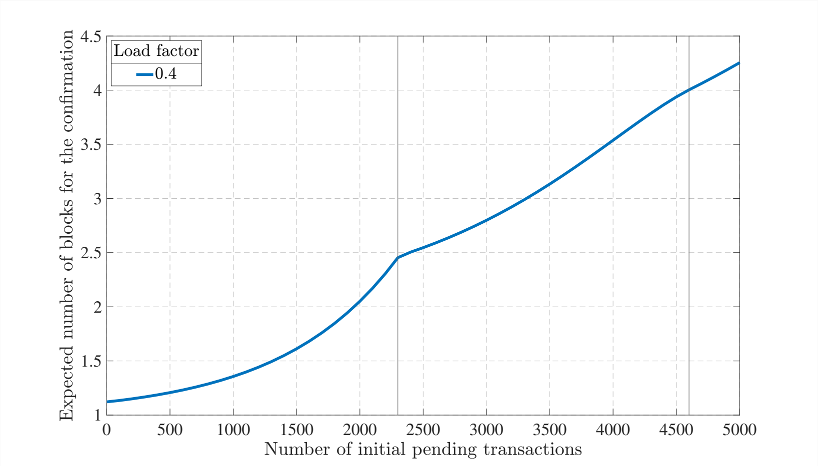
Expected number of blocks for the confirmation with different number of transactions in the MemPool as function of load factor

The impact of the load factor $\rho = 0.2$



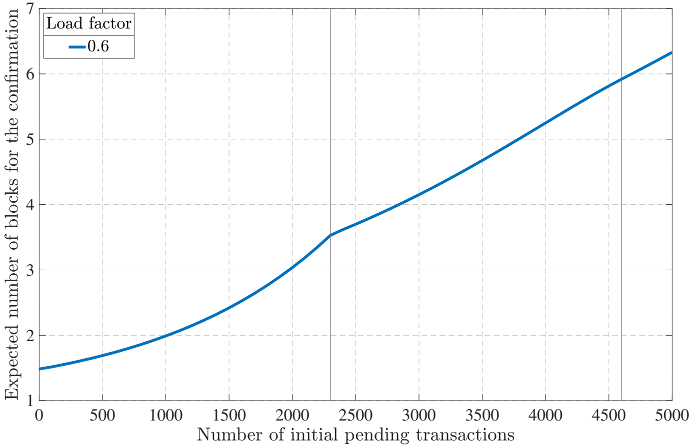
Expected number of blocks for the confirmation with different number of transaction for $\rho = 0.2$

The impact of the load factor $\rho = 0.4$



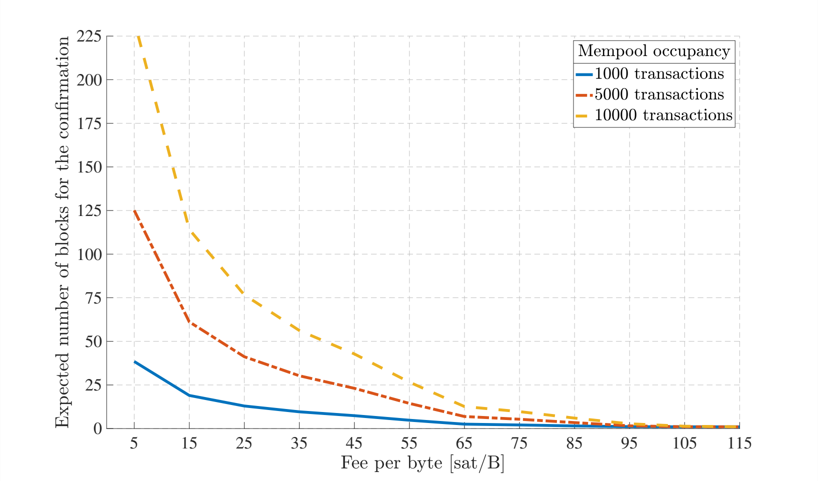
Expected number of blocks for the confirmation with different number of transaction for $\rho = 0.4$

The impact of the load factor $\rho = 0.2$



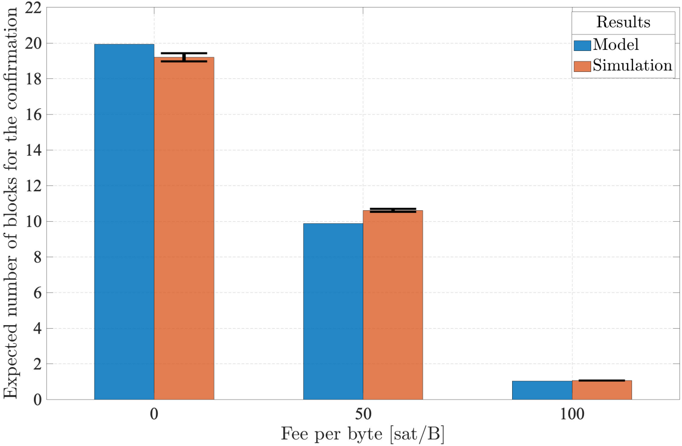
Expected number of blocks for the confirmation with different number of transaction for $\rho = 0.6$

The impact of the offered fee in heavy load



Expected number of blocks for the confirmation as a function of fee per byte in heavy workload conditions

Validation of the Model via trace-driven Monte Carlo simulation



Expected number of blocks for the confirmation as function of fee per byte for model and simulation results with $Y = 6,000$ and $\lambda = 3.21$ transactions per second

Conclusion

- We solve the transient problem of the $M/M^B/1$ queueing system
- We use it to study the confirmation time of transactions conditioned to:
  - Initial Mempool occupancy (this cannot be present in a stationary analysis)
  - Offered fee
  - Intensity of the workload
  - Distribution of the offered fees
- Good accuracy in the prediction especially for fast transactions

Future directions

Future directions

- How do we handle long-term predictions?
  - The arrival process is not time-homogeneous any more
- How do we embed the model in a game-theoretical framework?
  - Transaction fees adapt to the varying conditions
- How do we estimate the probability of dropping in the user-centric scenario?