NFPs 2: Performance

Mark Staples
Email: mark.staples@data61.csiro.au

www.data61.csiro.au

Outline

- Performance
- Latency
  - Transaction Inclusion in Public Blockchain
  - Automating Process Execution on Public Blockchain
- Throughput
- Summary

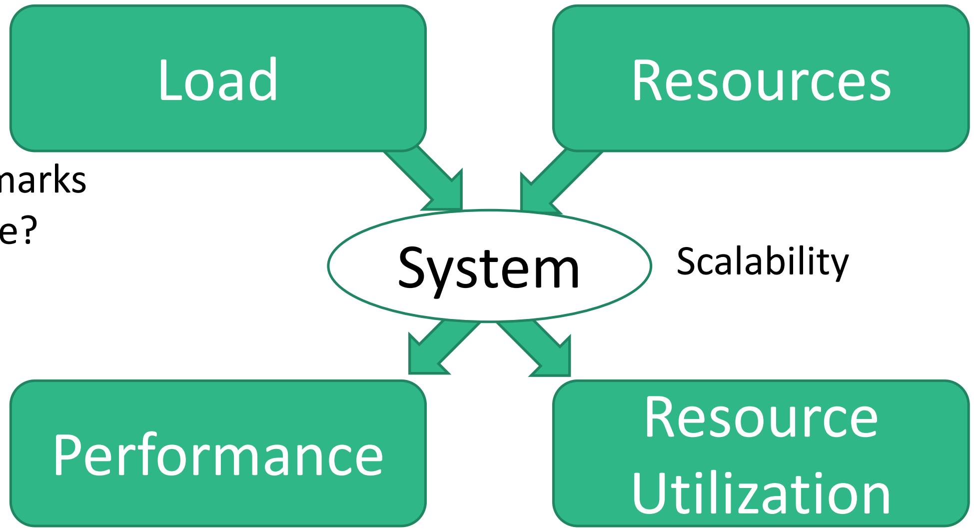
Performance

ISO/IEC 25010:2011 Quality Model

- **Functional suitability**
  - Functional correctness
  - Functional complete-ness
  - Functional appropriate-ness
- **Performance efficiency**
  - Capacity
  - Resource utilization
  - Time behavior
- **Compatib-ility**
  - Interop-erability
  - Co-existence
- **Usability**
  - Operability
  - User error protection
- **Reliability**
  - Availability
  - Recoverab-ility
  - Maturity
  - Fault tolerance
- **Security**
  - Integrity
  - Confidential-ity
  - Non-repudiation
  - Accountab-ility
  - Authent-icity
- **Maintain-ability**
  - Modularity
  - Reuseability
  - Modifiab-ility
  - Testability
  - Analyzab-ility
- **Portability**
  - Installability
  - Replace-ability
  - Adaptability

Performance-Related Considerations

Background
Sustained
Increasing
Stress
Standard benchmarks
Representative?
Production
...



Network
Memory
CPU
Disk
...

Scalability

Latency
Throughput
Timeliness
...

Capacity
Bottlenecks

Efficiency

Some Important Varieties of Performance

- Latency
  - How quickly does the system respond to a request?
- Throughput
  - How many requests can the system process in a fixed unit of time?
- Timeliness
  - Will the system always process a request in a specified time window?
- Scalability is an orthogonal issue
  - How does <*performance*> change under increasing load & system resources?
    - (Can also be interested in response when decreasing resources, under variable load)

To Understand Performance, You Need Measurements

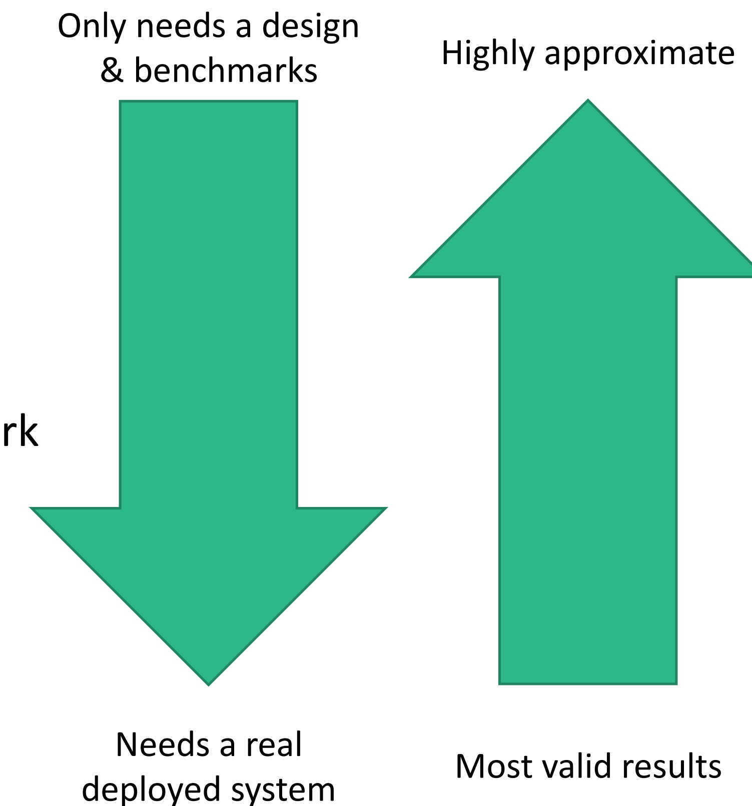
- Application-Level measurements
  - Overall performance; check requirements met; validate results from detailed models
- Component-Level measurements/ Micro-benchmarking
  - Piece-wise drivers of performance; observe interactions; identity bottlenecks

- Load testing – check system response & utilisation under increasing load
- Stress Testing – check maximum capacity; find resource bottlenecks
- Soak Testing – check system response under long-duration load

- "Benchmarking crimes"
  - https://www.cse.unsw.edu.au/~gernot/benchmarking-crimes.html

Predicting Performance?

- Analytic formulae
  - Highly abstract/simplified models
  - Quick & dirty
  - Only needs design & component benchmarks
- Simulation
  - Uses a model; many abstractions/simplifications
  - Only needs a design & benchmarks, but takes work
  - Starts to explore variation & interactions
- Monitoring
  - Needs real deployed system
  - Needs a monitoring framework
  - Most valid results; e.g. actual effects!



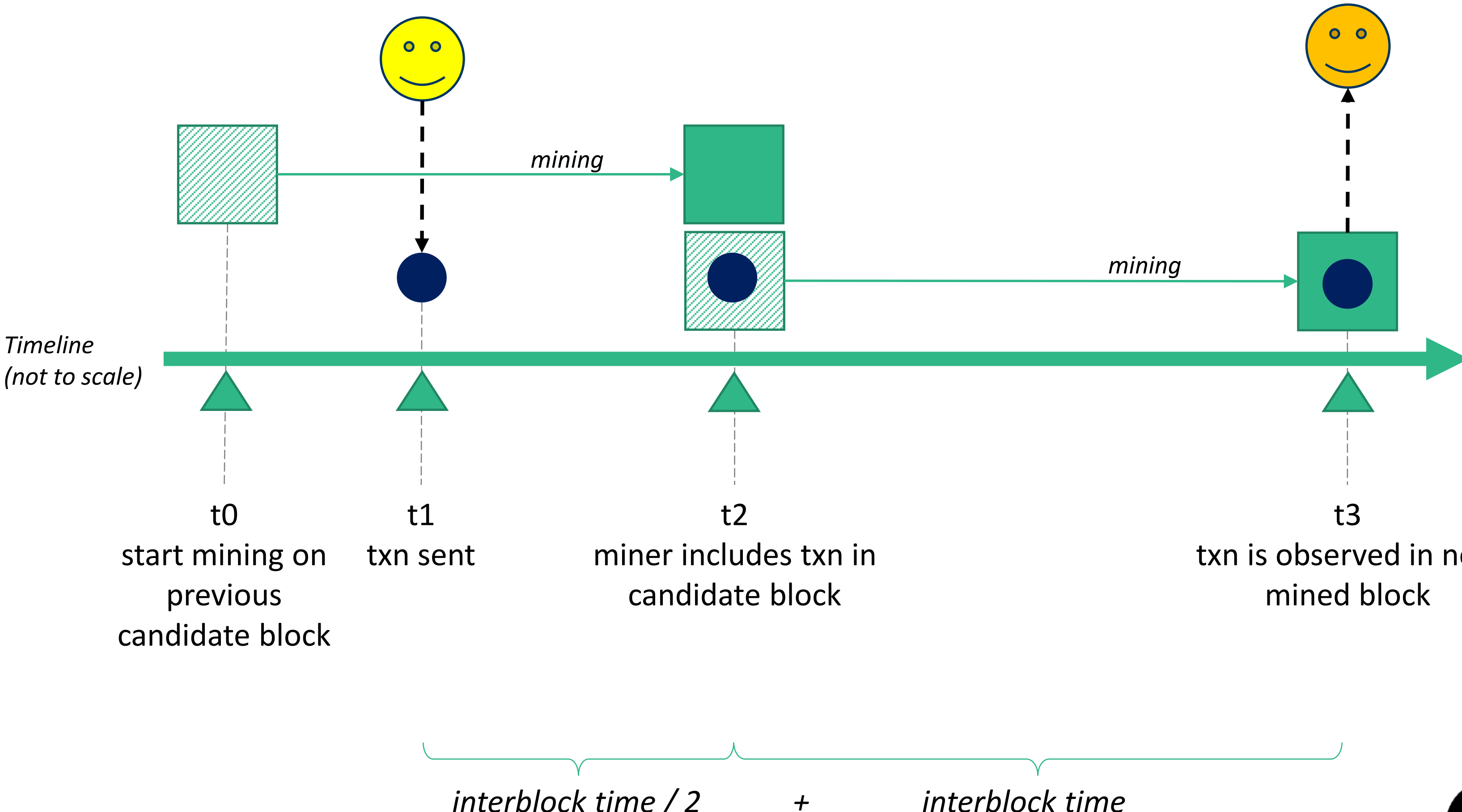
Some Resources in Blockchains

- For public blockchains
  - Cryptocurrency for Transaction Fees
    - Can affect probability of transaction inclusion (impacts transaction latency)
  - Mining difficulty for Proof of Work
  - Gas & Block Gas Limits (e.g. in Ethereum)
    - More of an Availability problem as discussed next week
  - Network capacity including speed of light for global transaction & block propagation
    - Impacts consensus for transaction inclusion latency
  - Transaction throughput capacity
    - Latency will be impacted when system is overloaded beyond capacity

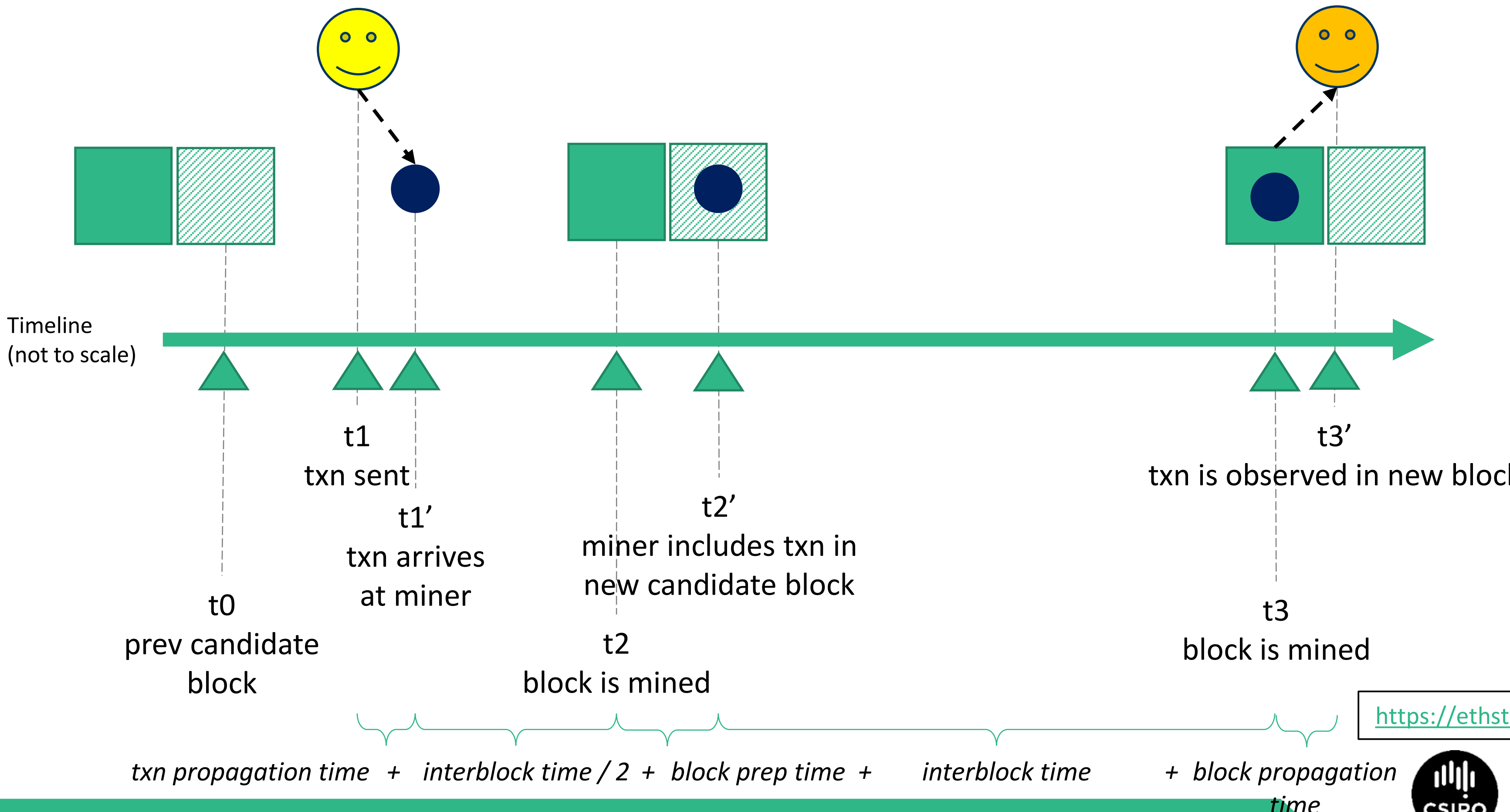
- For private blockchains
  - Normal cloud/enterprise server & network resource considerations

Latency

Transaction Inclusion Time in Public Blockchain…

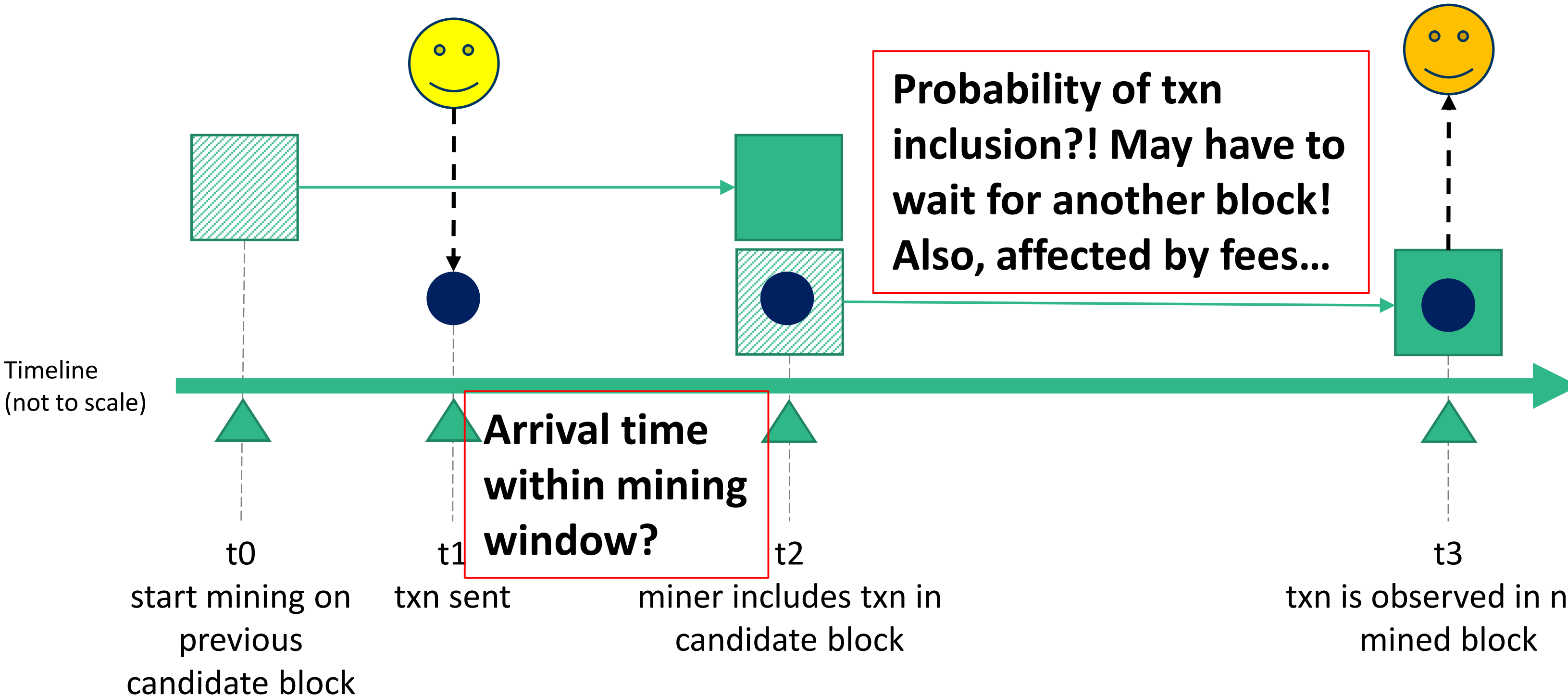


… Plus Extra Small Bits of Time



https://ethstats.net/

… Also, Major Sources of Variation in Latency

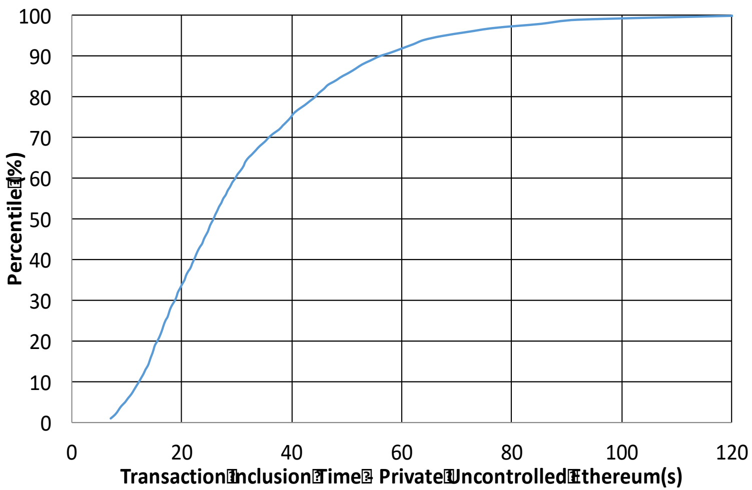


Previous interblock time?

Current interblock time?

Upshot is High Variation of Latency

- Not 1.5 x interblock time; in practice more like ~1.9 x inter-block time?
- But with a lot of variation! Some caused by txn fee, gas fee, etc
- You need to design your system to cope with this to meet requirements



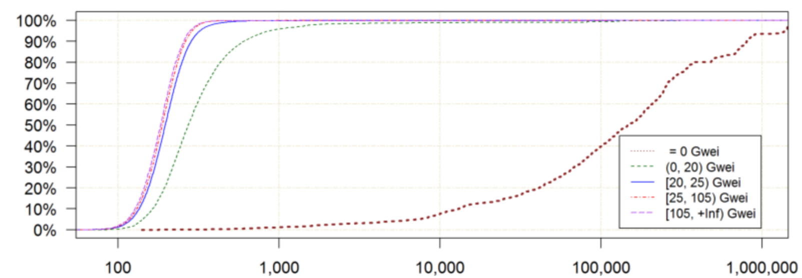
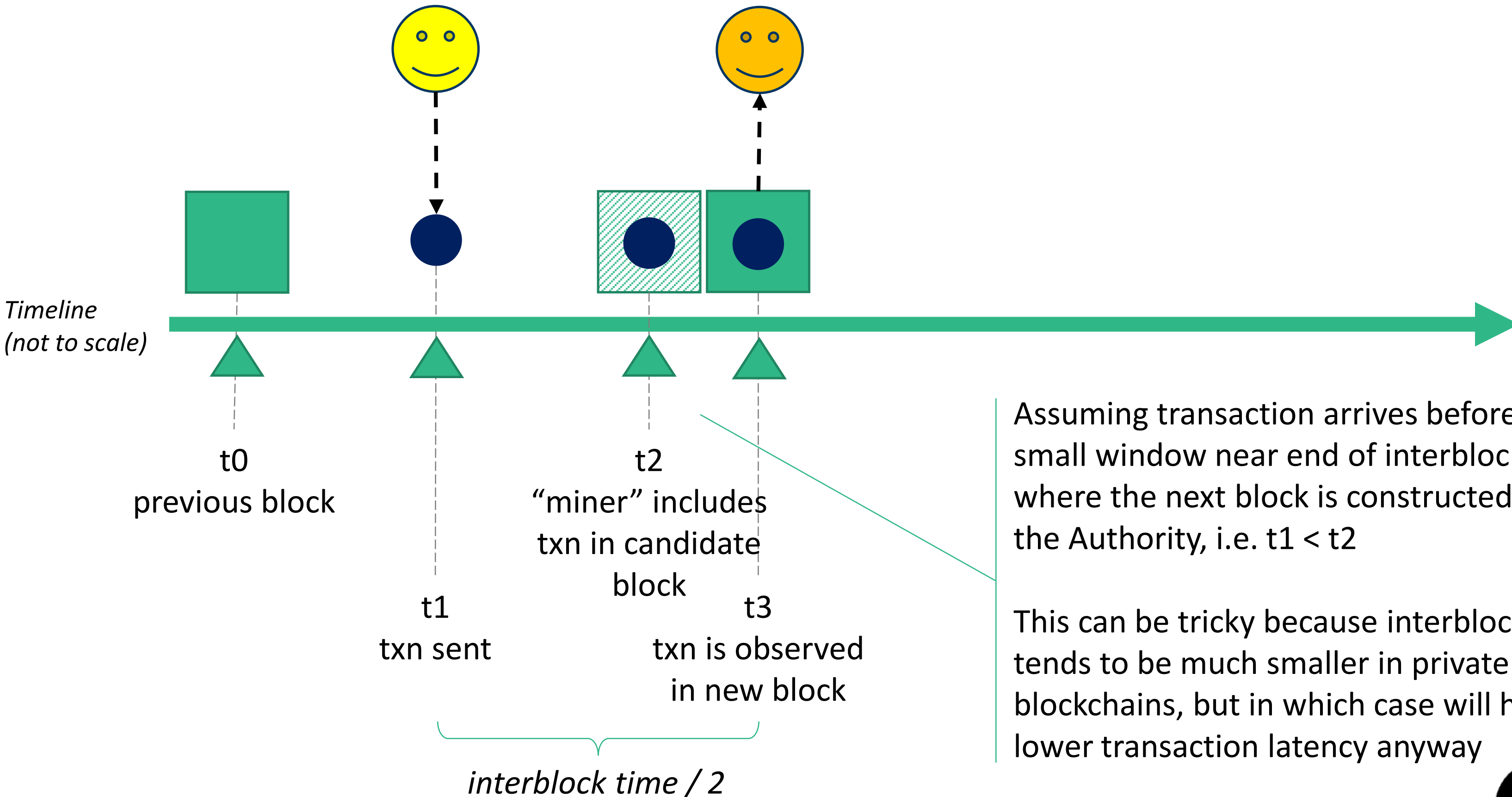


Figure 6: Commit delay (sec) for transactions based on gas price. Note logarithmic x axis.

N.B.: with 11 “confirmation blocks” adding a large “constant”-ish factor on the main cause of difference in latency variation

An Aside: Transaction Inclusion with Proof of Authority in Private Blockchain



Assuming transaction arrives before the small window near end of interblock time where the next block is constructed by the Authority, i.e. t1 < t2

This can be tricky because interblock time tends to be much smaller in private blockchains, but in which case will have lower transaction latency anyway

In Public Blockchains, Also Wait for Confirmation Blocks!?

- Nakamoto consensus can create “uncle” blocks
  - Blockchains using Proof-of-Work, Proof-of-Stake, …
  - Transaction you saw in a block turns out not to have been officially included in the blockchain
  - Only probabilistic, long-run, transaction inclusion

- Can increase confidence your transaction is really included, by waiting for subsequent “confirmation blocks”
  - For Bitcoin, often people say wait 6 blocks
  - For Ethereum, often people say wait 12 blocks
    - (Lower interblock time can increase likelihood of uncles)

- Waiting for confirmation blocks increases latency, and increases latency variability

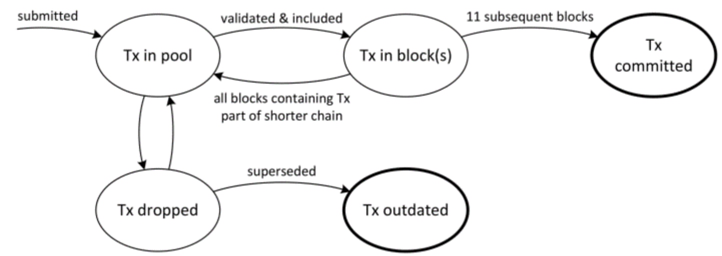


Figure 4: State machine for an individual transaction

Confirmations: Inclusion Confidence vs Latency

- Should be risk-based: how much confidence? vs. how much latency?

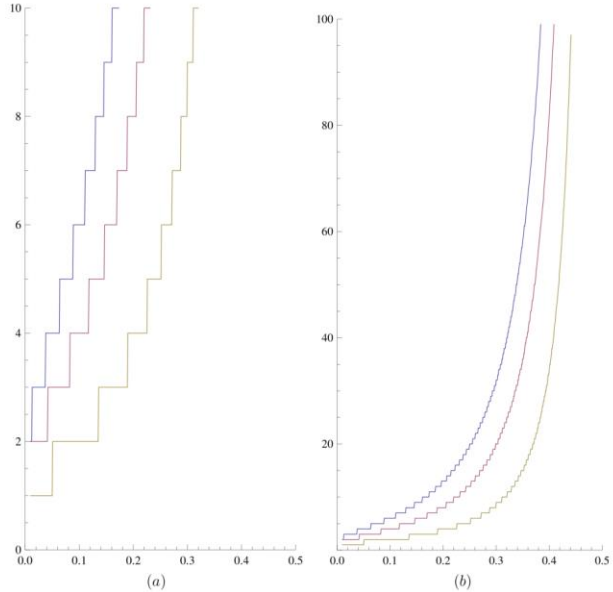


Figure 5: The number of confirmations required to keep the probability of success low, as a function of the attacker's hashrate, for various values of the target probability: 10% (yellow), 1% (purple) and 0.1% (blue). The graph is shown in two different vertical scales.

"Analysis of hashrate-based double spending"
https://arxiv.org/abs/1402.2009

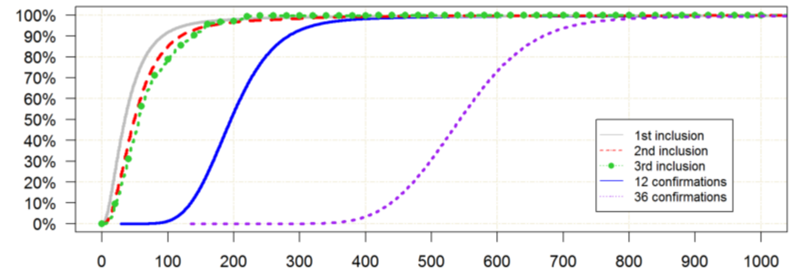


Figure 5: Time (sec) for first inclusion and commit (12 or 36 confirmations), as well as second and third inclusion of transactions that were previously not included in main chain.

"On Availability for Blockchain-Based Systems"
https://research.csiro.au/data61/wp-content/uploads/sites/85/2016/08/OnAvailabilityForBlockchain-BasedSystems-SRDS2017-authors-copy.pdf

不同的结构对latency的影响
1.gas控制结构
降低脚本执行复杂度,latency减小
2.选择不同的consensus algorithm

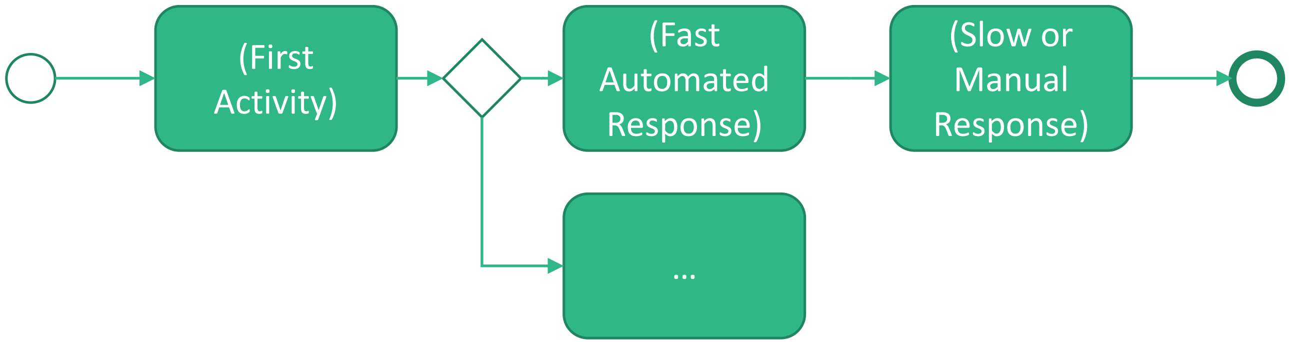
How Architects Can Influence Transaction Latency

nakamoto consensus 有孤块出现的可能,虽然被include了,但也有可能被其他链超越

- Offer a Transaction fee that meets/exceeds miners' expectations
  - Ethereum "fee" is gas price * gas used

  控制复杂度,复杂度高的话会影响latency
- Choose smallest number of confirmation blocks that is "safe" for your application
- Use another blockchain (private? or a public blockchain that works "better"?) 降低叔块出现的概率
  - Configure blockchain to use other consensus algorithms (PBFT, PoA, …)
    - Avoid Nakamoto consensus & possibility of uncles, so you can avoid confirmation blocks
  - Configure blockchain to target a small interblock time
- Work off-chain, e.g. using "state channel" design pattern, to be discussed in later lectures

An Example:
Process Execution on Public Blockchain

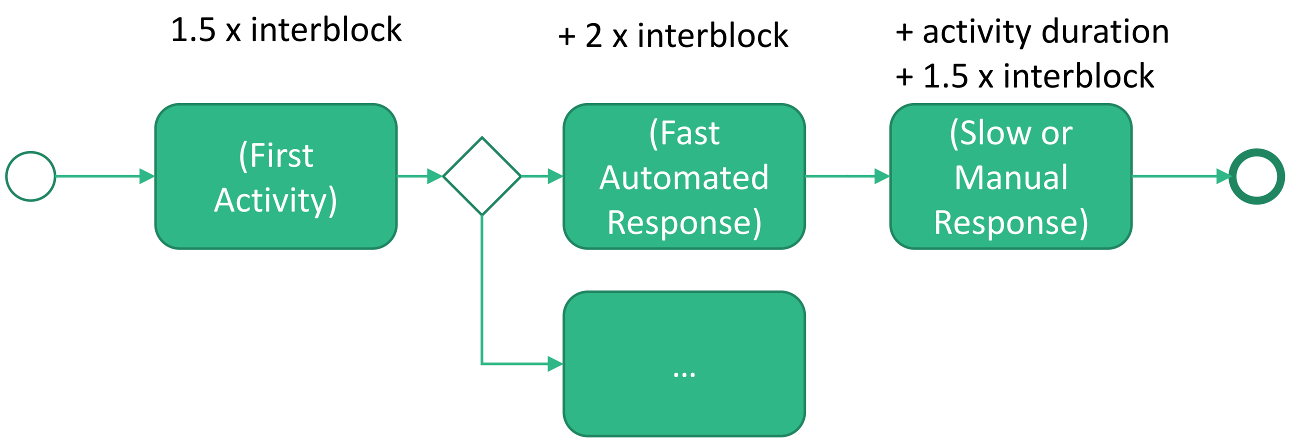


A process instance can be represented as a smart contract on a blockchain.

Each activity in the process can be a different function in the smart contract API.

An off-chain “trigger component” on a cloud or enterprise server watches a blockchain node and invokes the next activity in the process. (Immediately if activity is automated, otherwise e.g. in response to some manual activity.)

Latency of Process Execution on Public Blockchain



Here, the design does not use confirmation blocks.

Quick & dirty!
Assumes averages (median or mean)!

- Transactions arrive in the middle of an interblock time window
- Uniform average interblock time

The first activity will arrive in "the middle" of some interblock time window, so takes 1.5 x interblock times.

If activity response is very fast (in the same interblock time window), it will still be too late to include in the next block, so will take 2 x interblock times.

If the activity response is very slow (longer than the interblock time window), it will arrive in "the middle" of some future interblock time window

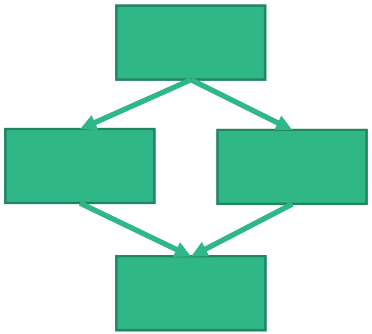
Simulation for Latency Estimation

- e.g. using Palladio
  - Open source, Component-Based
    - https://www.palladio-simulator.com/home/
  - Widely used for performance modelling and simulating
  - UML-like notation

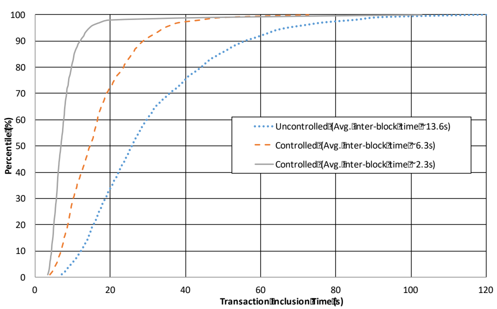
- Need to Benchmark Transaction Inclusion on Blockchain
  - Treat the blockchain as a black box – txns in/txns out
  - Measure time between submitting a transaction and receiving verification by a given number of confirmation blocks
    - Not just averages; measure the distribution of times
  - Also benchmark trigger implementation that invokes txns

"Predicting latency of blockchain-based systems using architectural modelling and simulation"
https://research.csiro.au/data61/wp-content/uploads/sites/85/2016/08/2017-ICSA-Blockchain-Latency-Sim-authors_copy.pdf

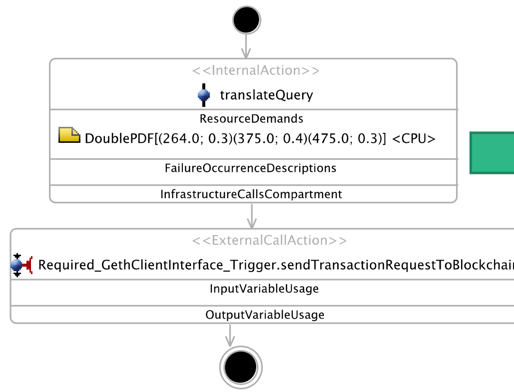
Simulation for Predicting Latency



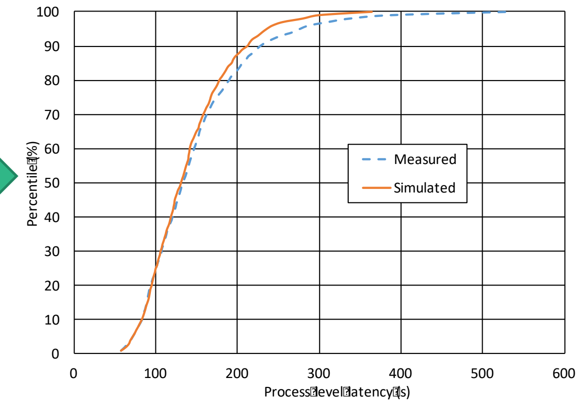
Architecture



Micro Benchmarking



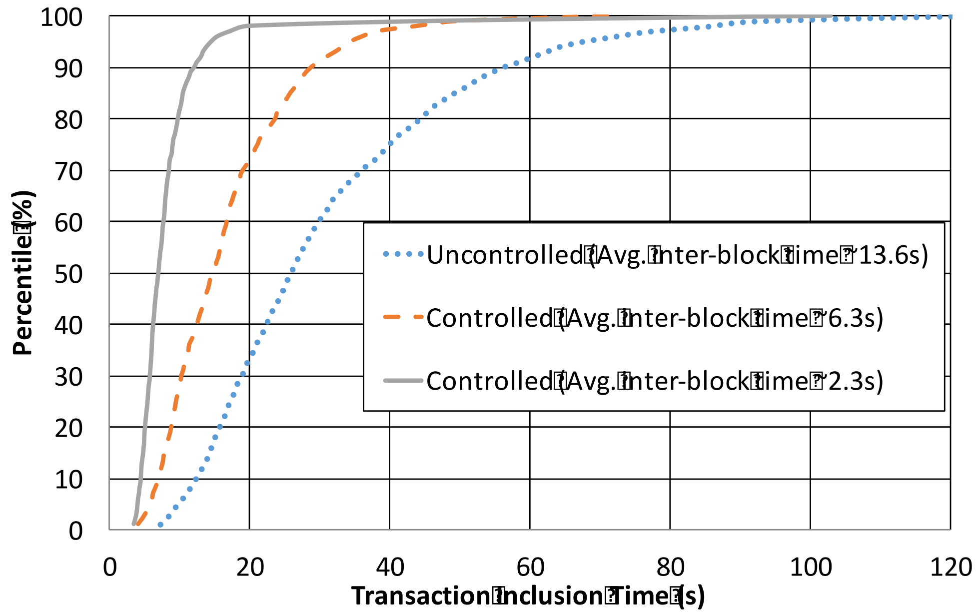
Model Construction



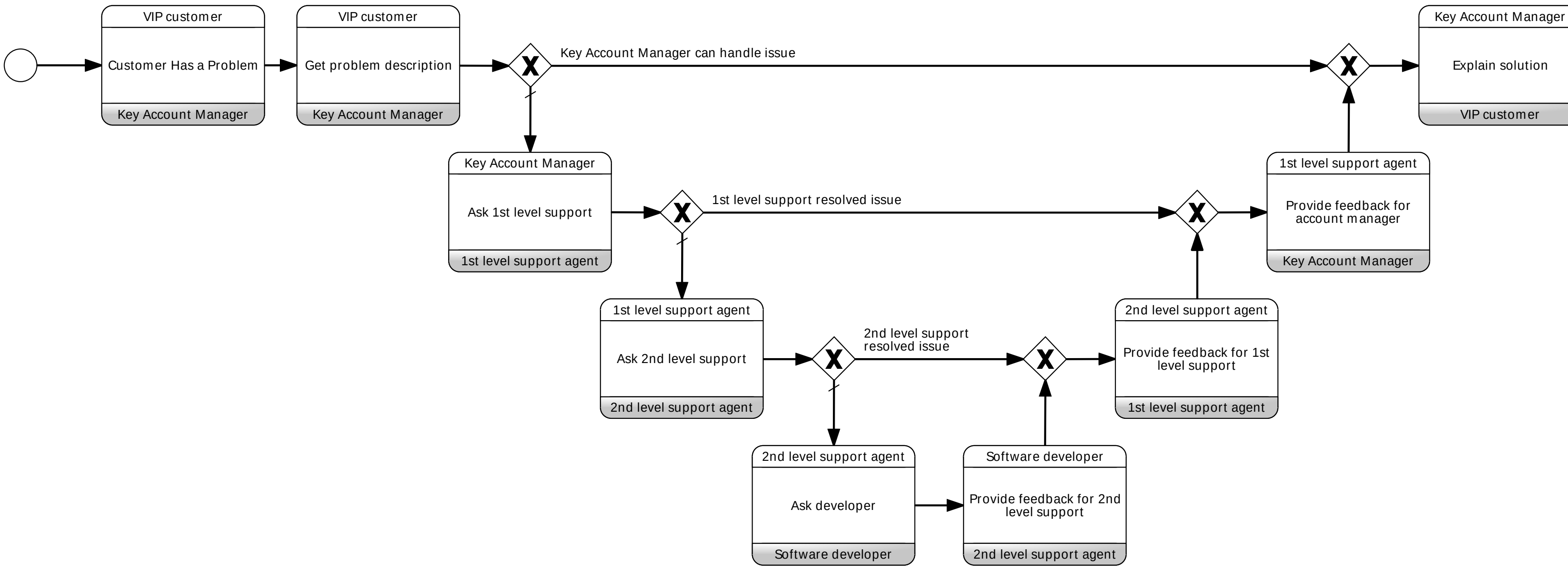
Simulation

Benchmarking Transaction Commit Time

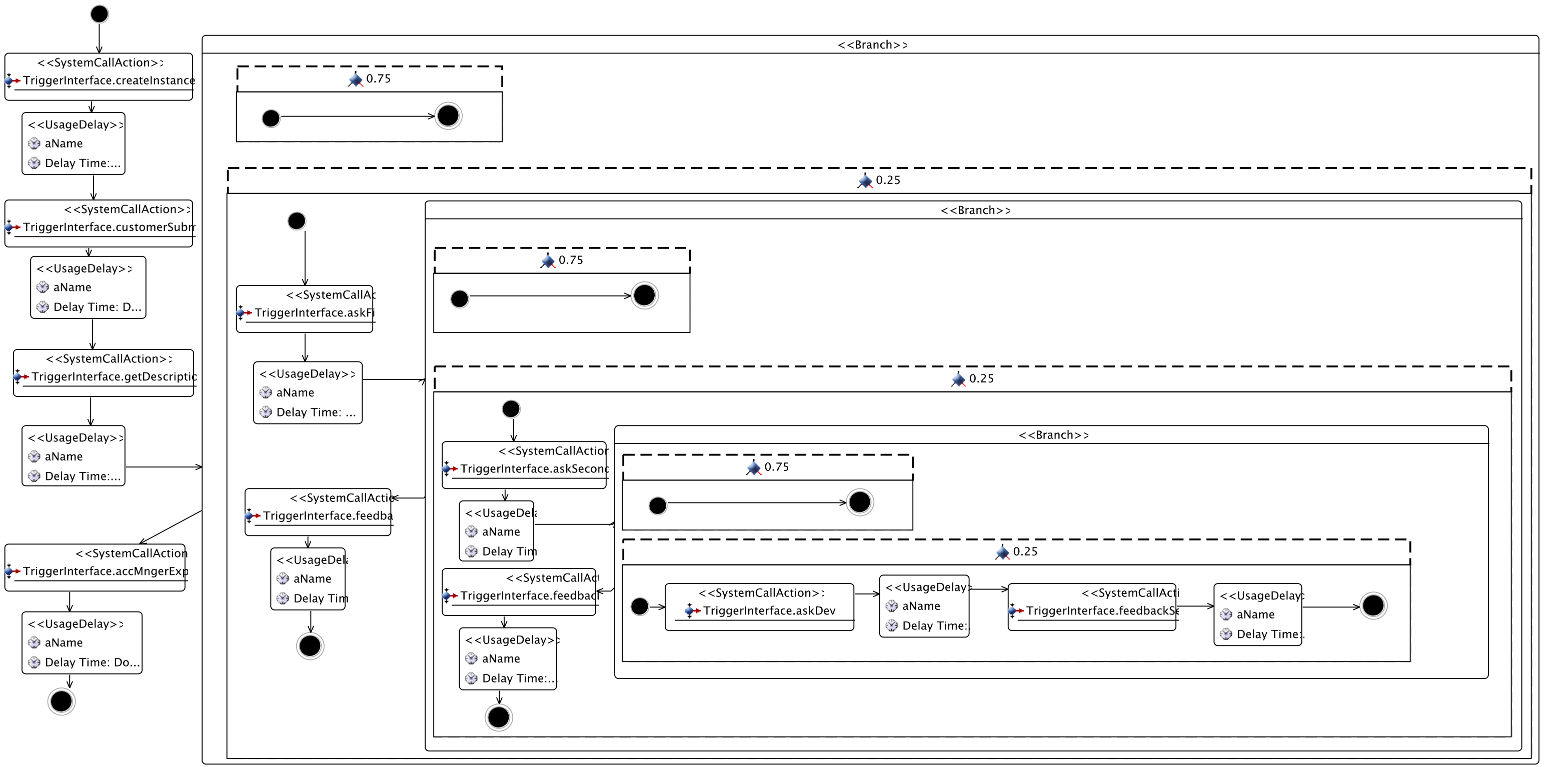
- *<u>Transaction Inclusion Time</u>*: Time Taken from Submitting transaction until transaction get included in blockchain



Example Incident Management Business Process



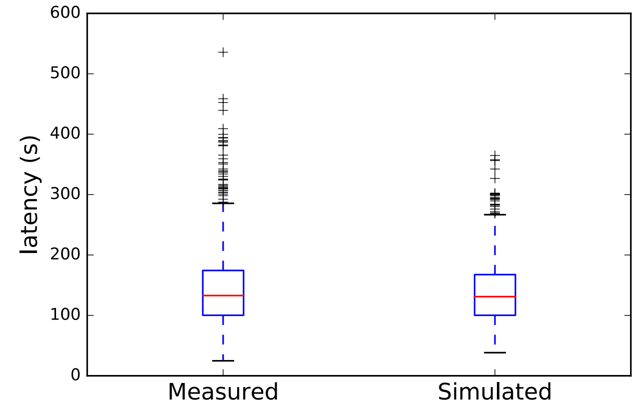
Example Usage Model

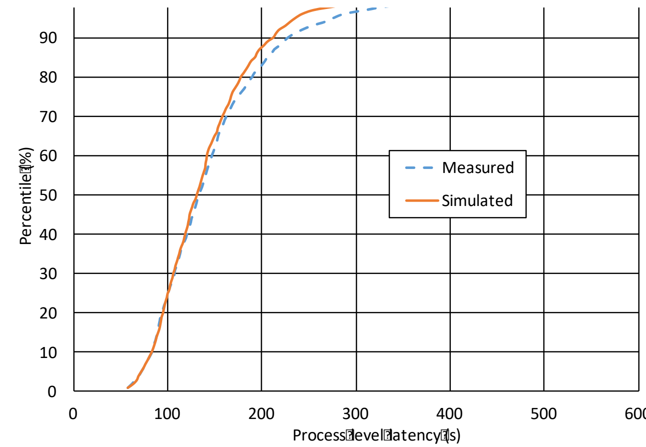


Usage diagram

Example Results & Checking vs. Real System

	Measured	Simulated	Relative error
Median latency	136s	134s	1.6%
Standard error of median (SEM)	1.27	1.07	-
95th Percentile	274s	248s	9.4%
99th Percentile	373s	329	11.5%





Throughput

Throughput

- How many transactions can you process in a unit of time?

- e.g. Visa card processing, 2000 tps daily average, peak capacity ~60000 tps
  - https://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html

- Bitcoin has poor throughput capacity (3 to 7 tps)
- Ethereum has poor throughput capacity (7 to 25 tps)
  - https://medium.com/coinmonks/understanding-cryptocurrency-transaction-speeds-f9731fd93cb3

- You as an individual user are unlikely to create system overload!
  - But, you need to understand trends in ecosystem utilisation and bottlenecks
    - e.g. At one point in time, about a third of transactions on Ethereum were for cryptokitties

Throughput Limits

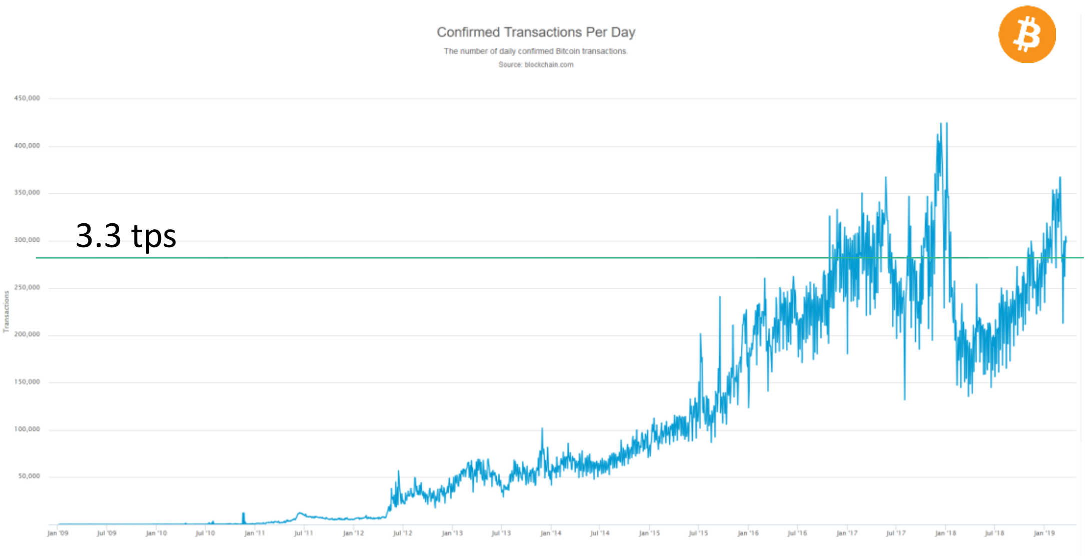
- Transactions per block * blocks per second = Transactions per second

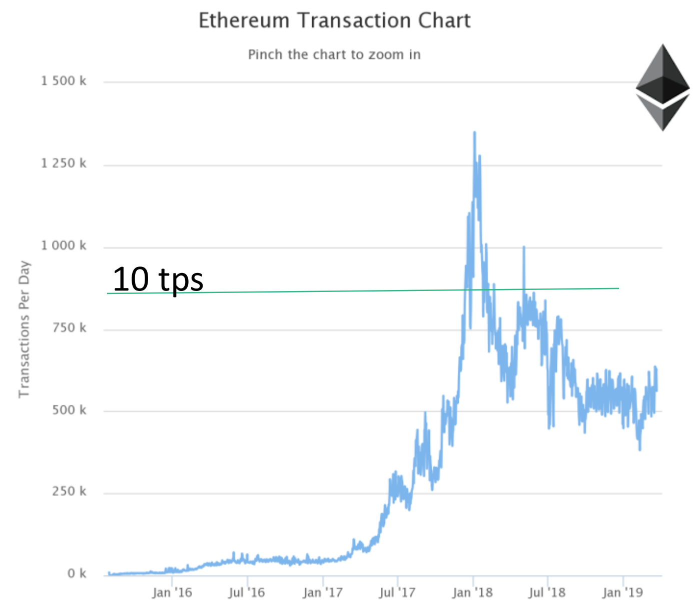
- Bitcoin limited by block size, and average transaction size
  - 1000000 Bytes / 495 Bytes / 600 seconds = 3.37 tps
    - (growth in use of segwit and other txn size optimisation can double? throughput)

- Ethereum limited by block gas limits, and average transaction gas
  - 8000000 gas block limit / 76000 gas / 15 seconds = 7 tps

- Limits are there mainly to control block propagation times
  - To control likelihood of uncles, DDoS, and control centralisation of mining

Growth in Throughput





Red Belly Blockchain Throughput

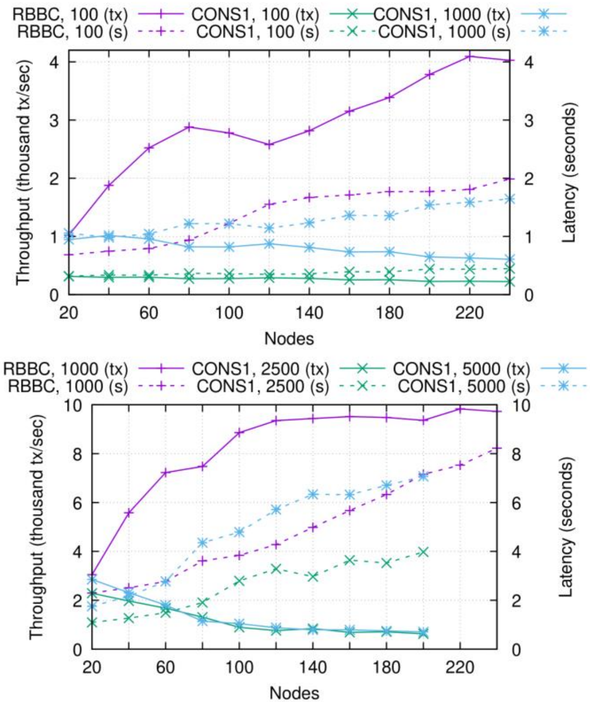


Figure 3. The performance of CONS1 and RBBC with $t + 1$ proposer nodes; the number following the algorithm name represents the number of transactions in the proposals; solid lines represent throughput, dashed lines represent latency

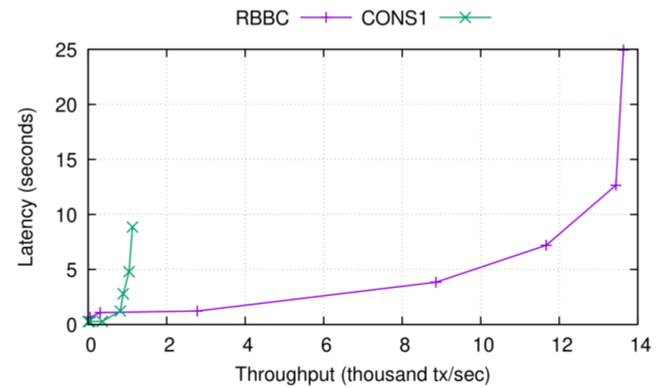


Figure 4. Comparing throughput and latency of CONS1 and RBBC with $t + 1$ proposer nodes on 100 geodistributed nodes; each point represents the number of transactions in the proposals, either 10, 100, 1000, 2500, 5000 or 10000

"Evaluating the Red Belly Blockchain"
http://arxiv.org/abs/1812.11747

DAGs, Sharding, etc

- Blockchains enforce a strictly ordered single list of transactions
  - Critical for integrity of transfer of digital assets (no “double spend”)
  - But creates a throughput bottleneck

- DAGs (“tangles”), e.g., IOTA, …
  - Not a list, but a directed acyclic graph
  - Multiple heads (“tips”), each processed mostly concurrently, repeatedly diverging/merging
  - Hard to maintain data integrity across the heads – transactions can conflict!
    - Can, e.g. resolve a bit like like Nakamoto consensus – tentatively included until merged

- Sharding, e.g. future Ethereum?
  - Split chain into smaller chunks (horizontal partitioning), each processed mostly concurrently
    - Might increase Ethereum throughput by 1000x
  - Hard to maintain data integrity across the shards
    - Shards interact with a protocol only when necessary, to maintain global integrity

- Networks or hierarchies of interrelated blockchains
  - e.g. geographically-sharded IoT blockchain networks

How Application Architects Can Influence Transaction Throughput

- Use another blockchain (private? or a public blockchain that works "better"?)
  - Configure blockchain to use other consensus algorithms (PBFT, PoA, …)
  - Sharding, DAGs
  - Configure blockchain to use a larger block size, or smaller interblock time
  - If you don't need exchange of digital property, consider networks of blockchains, DAGs, …

- If you have to use a specific public blockchain, then focus on how you use the blockchain, and focus on other areas of the application architecture
  - Reduce the data you put on-chain (sampling, digests, hashes, …)
  - Do more work off-chain, e.g. using "state channel" design pattern, to be discussed in later lectures

Summary

Summary: General

- (Load × Resources) → (Performance × Resource Utilisation)
  - Measurement is critical to understand all this in practice

- Kinds of Performance: Latency vs Throughput (vs Timeliness)

- Blockchains have high latency and high variability of latency
  - Variability is significantly affected by choice of consensus mechanism
  - Number of confirmation blocks are a risk-based decision
    - Can avoid them if you don't use Nakamoto consensus/probabilistic commits

- Blockchains have poor throughput
  - Affected by ledger structure and consensus mechanism
  - Do you need exchange of digital assets?
    - If you don't need a global integrity property, you can avoid major bottlenecks

Summary: Predicting Latency

- For a single transaction on public blockchain

  $$\text{latency} = 1.5 * \text{interblock time}$$

  - Assuming transaction is included ASAP
  - Ignoring transaction/block propagation times, and block preparation time
  - On average (ignoring variation in interblock time, arrival time)
- For sequence of n > 1 transactions on public blockchain

  $$\text{latency} = 1.5 * \text{interblock time} + (n-1)*2\ \text{interblock time}$$

  - Assuming next transaction is injected immediately into the interblock window
- Can use simulation for more precise prediction of application-level latency
  - Can explore latency variation, interactions between components, and bottlenecks
- Or, use measurement and test of real(istic) system under real(istic) load!

THANK YOU

www.data61.csiro.au