Reliability, distributed consensus and blockchain

COSC412

Learning objectives

- Encourage you to always design for failure
- Appreciate how decentralised consensus is useful to support principles of security such as reliability and non-repudiation
- Gain a high-level view of blockchain approaches and how they support, *e.g.*, applications like bitcoin, and other emerging decentralised autonomous systems

Securing valid results on fallible machines

- Digital devices suffer (non-malicious) failures
  - RAM corruption errors—c.f., ECC memory
  - Storage media may fade or malfunction
    - Beware cheap writable optical media or flash storage
    - SSD devices fail very differently from magnetic hard drives
- Also may have critical software fail:
  - filesystem bugs
  - compression library bugs
  - system use contrary to supported operation

One solution: rerun your computations

- If you can estimate the probability of failures, you can determine how many trials of a computation you need to achieve a given level of confidence in the result
  - Excessive system failures may become overshadowed by other concerns anyway…

- Of course multiple trials need not be run in serial:
  - can structure repeatability within a software service
  - cloud computing provides convenient elasticity for parallelism

Aside: machines designed to fail frequently

- Reliability of computers is adjustable
  - Can trade off against speed, power consumption, *etc*.
  - Consider the practice of overclocking CPUs:
    - may need to apply CPU voltage adjustments;
    - may affect reliability of computation—possibly catastrophically!

- Computer participates in a group repeating results?
  - Can purposefully design such a computer to be less reliable
  - May end up with a net saving in resource trade-off

Merkle tree: efficient integrity checking

- Consider a set of data blocks D_i, then:
  - A hash $H(D_i)$ is computed for each data block D_i
  - A tree is built, with parent hash hashing hashes of its children
  - The root hash will thus summarise all the data blocks

- Checking hash on particular D_i can be done cheaply
  - Get trusted root hash; other hashes can come from anywhere
  - Used within Bittorrent to check blocks retrieved build valid file
  - Also with ZFS, within bitcoin transaction blocks, *etc*.

Distributed consensus—trustworthy results

- Common in more than just storage systems, *e.g.*:
  - Master/master relational database server replication
  - NoSQL: *e.g.*, use of gossip protocols and eventual consistency
  - Network infrastructure such as routers with hot spares
- Systems now exist that just handle consensus gathering
  - *e.g.*, Apache ZooKeeper offers distributed synchronisation
- ZooKeeper used in other systems: Hadoop, HBase, …

Apache ZooKeeper

- Essentially a multi-server, key-value database system
  - However, emphasis is on correctness and synchronisation
- Has always been a core component of Hadoop
  - Helps coordinate & manage scheduling of map-reduce tasks
- Key property: facilitates atomic broadcast
  - Under atomic broadcast all correct processes in a distributed system receive the same sequence of events, or all abort

Add in potentially malicious parties

- Apache ZooKeeper commonly used when we trust all servers: they are owned by one organisation, on a LAN

- When malicious parties may be participating, the consensus set size must grow
  - Need a majority of votes from the **assumed-benign** server set

- Could we choose not to control the server set?
  - Enter blockchain, and bitcoin as an example of using it …

Warm up exercise: build a cryptocurrency

- How do we make a cryptocurrency "coin"?
- How do we identify coin owners?
- How can we protect the system from forgery?
- How do we record ownership and transfer of ownership?

- Can copy digital assets perfectly, so how can coins be single-use?

Distributed consensus needs within bitcoin

- To work, currencies need to track **who** has **what**
  - Normal currency uses TTPs such as mint, banks, *etc*.
- bitcoin has all validating nodes store the **whole ledger**
  - This distributed ledger indicates order of transactions
  - Collectively agreeing its content avoids double-spending
- A wallet is a hash of a public key a client generates
  - Own private key? Can prove your connection to transactions
    - … don’t actually need a representation of ฿ apart from ledger

Proof of work—validate ฿ transactions

- Must protect validation from Sybil attacks, so:
  - Make it computationally costly to incorporate new transactions
    - move to how much computing power you control, not just the number of identities that you control (*i.e.*, the basis of Sybil attacks)
  - Make it rewarding to incorporate new transactions—more later
- Validator collects transactions into a block
  - checks transactions internally first—could be double spending
  - forms Merkle tree over transaction hashes
  - to close off the block, it applies proof of work algorithm

bitcoin transaction validation

- Proof of work must be easy to check; hard to compute
  - In some ways like a hard-to-apply digital signature
- In bitcoin, must find a nonce that when appended to the block of transactions+ gives a number less than a target
  - SHA-256 hash function used, specifically
  - Target is dynamic: ensures blocks take ~10 minutes to compute, regardless of changes in net computational resources available
- Block**chain** because block hash included in next block
  - September 2020: bitcoin blockchain is about 298 GB

 + and other things we are not talking about

Validators, mining, fees and the network

- Bitcoin miners are carrying out validation of blocks
- Two incentives for miners to solve block hash task:
  - payment of 6.25 bitcoin since May 2020 (was 50₿ in 2009!)
    - value halves periodically; by 2140 CE no further bitcoin increase
  - ability to levy fees—commercial competition applies

- Broadcast communication between miners uses a peer-to-peer protocol
  - avoids central infrastructure… and knowing the miner set (!)

Results from block validation

- Rate is ~10 minutes, but this is probabilistic
  - *e.g.*, might guess an appropriate nonce first off (if really lucky)

- Automatically helps serialisation: variance in mining time is larger than the message broadcast time
  - Miners want to publish results ASAP so to receive payment
  - (Some potential attacks do involve holding back a solution.)

- Still possible for multiple answers to be broadcast, so…

Blockchain forks

- When nodes hear multiple solutions they keep them all
- Subsequent mining is only done on your longest fork
  - Extremely unlikely that parallel forks will continue for long
    - (Well, unless fork is due to a software bug, which has happened…)
  - Probability distribution likely to clearly favour one branch
- Attacker with significant resources can try to keep fork alive, but cost, coordination and probability won't help
  - (Some attacks involve late revealing of privately mined forks.)

How/when is a transaction approved?

- Clearly the transaction has to be recorded in a block
- Two simple rules are applied:
  - Relevant block must be in the longest fork of blockchain
  - Five or more blocks must already follow it in the blockchain
- This causes a transaction clearing delay (in effect)
  - Consider possible attacks, *e.g.*, partitioning of network
  - Probably impractically difficult to effect

Content of transactions

- No persistent coins: serial numbers are transaction hashes
- Transaction specifies a number of inputs and outputs, with inputs usually previous transactions
  - can output back to yourself, thus pocketing ‘change’
  - remainder of input, after subtracting output, is transaction fee
- Since all transactions are in the blockchain:
  - can search back in time to find transaction:
  - either genesis block (50 bitcoin) or a coinbase mining reward

Nodes in bitcoin network

- There are four main roles nodes can take on:
  - **Network**—all nodes help routing within the p2p protocol
  - **Wallet**—manage keys that show ownership of transactions
  - **Miner**—participate in the proof-of-work block verifications
  - **Blockchain**—can carry the full blockchain
- Bitcoin Core reference client contains all four functions
  - Miners may leave out wallet
  - Lightweight wallet only has wallet and network components
  - Some notes may store blockchain, but not do mining

Many more aspects of bitcoin not discussed

- bitcoin blocks also include management parameters:
  - *e.g.*, version numbers to allow the protocol to be modified
  - Versioning is very important given that the protocol behaviour is the fundamental basis on which the currency is built
- bitcoin specifies transactions with a scripting language
  - P2PKH—"pay to public key hash" is a common transaction
  - "multisig" transactions allow m-of-n public key sign-off
  - Smart contracts can be encoded, beyond money transfer

bitcoin scalability challenges

- Originally, blocks had no size limit, but that risks DoS
  - Added a limit that blocks can only be 1 megabyte at most
- Blocksize limit has caused scalability problems:
  - Provides for about three transactions per second at best
  - Ten minutes to add a block to blockchain
  - Thus bitcoin transactions can take hours to confirm
- Segregated Witness (SegWit) approx. doubles size

Concerns: anonymity, privacy and value

- bitcoin has been discussed as being anonymous
  - This makes little sense—the entire ledger is available publicly!
  - However it is true that public keys need not be identified
- Linkability concerns: metadata may allow subsequent determination of wallet's owners
  - Large state organisations likely want to do this,
    - e.g., law enforcement
- State players globally are key to bitcoin value

Blockchain aside from bitcoin

- Increasingly blockchain services are being offered independently of technologies such as bitcoin
  - Blockchain as a Service is offered on the commercial cloud
- There is much hype, and often gaps in understanding
  - However bitcoin helped show ways in which decentralised systems can appear to form distributed consensus
  - Many aspects existed already, such as in peer-to-peer systems
  - Many commercial organisations are interested…

Different sorts of blockchain systems

- **Permission-less** systems—bitcoin, Etherium, *etc*.
  - Need Proof of Work (bitcoin, Etherium), Proof of Stake (Nxt), …
- **Permissioned**—there is control over who participates
  - Can use algorithms like Paxos or RAFT to form consensus
  - … similar sorts of systems existed previously
- Other axis is **public** / **private**
  - sovrin is a permissioned+public blockchain managing identity
  - hyperledger is a premissioned, private blockchain

Non-currency blockchain uses

- Supply chain management: tracked asset transfer
  - Particular with respect to pharmaceuticals
  - Many organisations; common goal; fraud impractical
- Microgrids and neighbourhood electricity trading
- e-democracy and voting (how could that go wrong?)
- Always ask: is blockchain really needed? Alternatives?

Blockchain 2.0

- Executable contracts rather than transfer of currency
  - bitcoin already shows practicality of scripting language
  - bitcoin facilitates agreement of future events (& cancelation)

- Example applications cover legal, financial, *etc.*—
  - Systems for royalty collection on behalf of performers
  - Decentralised social networks; gaming; gambling
  - Conveyancing; finance; insurance
  - Government record storage; electronic health records …

Ethereum

- Ethereum aims to build a global computing platform
  - Cannot be shut down easily
  - Can scale up and down
  - Resistant to censorship and other interference

- Ethereum Virtual Machine
  - Platform on which code executes

- Usually need some sort of bridge to other web APIs

Blockchain scheme governance

- What if a protocol vulnerability is discovered?
  - Say a hacker runs away with credit with millions of dollars
  - Entire blockchain system can agree to rewind history
  - … but this is a capability blockchain systems seek to give up

- Etherium e.g.: Decentralized Autonomous Organization
  - Raised $150m crowd-sourced funding; DAO was ~15% of ether
  - Code had vulnerabilities; hacker siphoned off a third of DAO
  - Soft-fork and hard-fork resolutions discussed; hard-fork done

Conclusion

- Failures can threaten security by affecting availability
  - Hardware and software problems
- Efficient means exist to reach decentralised consensus:
  - Merkle trees for checking integrity
  - Apache ZooKeeper, within a known set
  - Proof-of-work within blockchain schemes such as bitcoin
- Discussed how bitcoin works despite threats
- Outlined possible future blockchain applications