Philadelphia Ethereum Blockchain Meetup reading group discussion on Polkadot: Vision for a Heterogeneous Multi-chain Framework

Draft 1

by Gavin Wood (Ethereum & Parity)

February 21, 2019

Obtain these slides at:
https://github.com/smoelius/polkadot-slides/blob/master/PolkaDotSlides.pdf

Abstract

- Present-day blockchain architectures all suffer from a number of issues not least practical means of extensibility and scalability.
- We believe this stems from tying two very important parts of the consensus architecture, namely **canonicality** and **validity**, too closely together.
- This paper introduces an architecture, the **heterogeneous multi-chain**, which fundamentally sets the two apart. ...
- The heterogeneous nature of this architecture enables many highly divergent types of consensus systems interoperating in a trustless, fully decentralised "federation"...

[*Preview*]

[*Figure 2, pp. 6*]

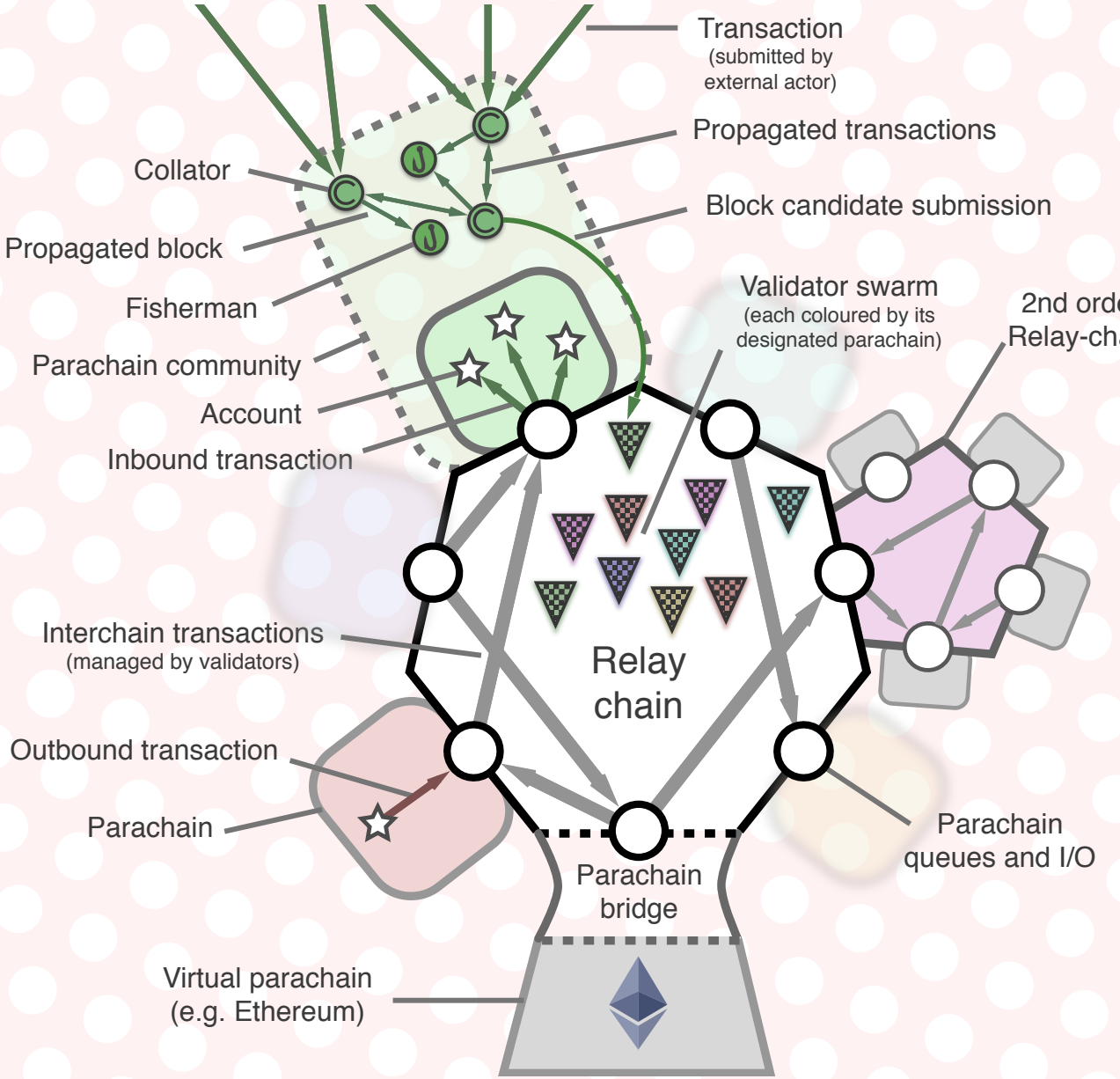

Figure: A summary schematic of the Polkadot system. ...

[*Figure 2 full caption*]

- A summary schematic of the Polkadot system.
- This shows collators collecting and propagating user-transactions, as well as propagating block candidates to fishermen and validators.
- It also shows how an account can post a transaction which is carried out of its parachain, via the relay-chain and on into another parachain where it can be interpreted as a transaction to an account there.

[*§5.4, pp. 11, just below Figure 3*]

- To ensure minimal implementation complexity, minimal risk and minimal straight-jacketing of future parachain architectures, these interchain transactions are effectively indistinguishable from standard externally-signed transactions.

[*End of Preview*]

[*Outline*]

1. Preface

- This is intended to be a technical "vision" summary of one possible direction that may be taken in further developing the blockchain paradigm together with some rationale as to why this direction is sensible. ...
- It is not intended to be a specification, formal or otherwise.
- It is not intended to be comprehensive nor to be a final design.
- It is not intended to cover non-core aspects of the framework such as APIs, bindings, languages and usage. ...

1.1. History

- 09/10/2016: 0.1.0-proof1
- 20/10/2016: 0.1.0-proof2
- 01/11/2016: 0.1.0-proof3
- 10/11/2016: 0.1.0

2. Introduction

- Blockchains have demonstrated great promise of utility over several fields including "Internet of Things" (IoT), finance, governance, identity management, web-decentralisation and asset-tracking.
- However, despite the technological promise and grand talk, we have yet to see significant real-world deployment of present technology.

[*Introduction is continued on the next slide.*]

2. Introduction [*continued*]

- We believe that this is down to five key failures of present technology stacks:

  Scalability How much resources are spent globally on processing, bandwidth and storage for the system to process a single transaction and how many transactions can be reasonably processed under peak conditions?

  Isolatability Can the divergent needs of multiple parties and applications be addressed to a near-optimal degree under the same framework?

  Developability ...

  Governance ...

  Applicability ...

- In the present work, we aim to address the first two issues: scalability and isolatability. ...

[*Introduction is continued on the next slide.*]

2. Introduction [*continued*]

- ...current real-world blockchain networks are practically limited to around 30 transactions per second. ...
- This is due to the underlying consensus architecture:
  - the **state transition mechanism**, or the means by which parties collate and execute transactions, has its logic fundamentally tied into
  - the **consensus "canonicalisation" mechanism**, or the means by which parties agree upon one of a number of possible, valid, histories.

  [*Emphasis added.*]

  ...
- It seems clear, therefore, that one reasonable direction to explore as a route to a scalable decentralised compute platform is to decouple the consensus architecture from the state-transition mechanism. ...

2.1. Protocol, Implementation and Network

- Like Bitcoin and Ethereum, Polkadot refers at once to a network protocol and the (hitherto presupposed) primary public network that runs this protocol. ...

2.2. Previous work

- Decoupling the underlying consensus from the state-transition has been informally proposed in private for at least two years—Max Kaye was a proponent of such a strategy during the very early days of Ethereum.
- A more complex scalable solution known as **Chain fibers**, dating back to June 2014 and first published later that year[1], made the case for a single relay-chain and multiple **homogeneous** chains providing a transparent interchain execution mechanism. ...
- Factom [Snow et al., 2014] is a system that demonstrates canonicality without the according validity, effectively allowing the chronicling of data. ...

[1] https://github.com/ethereum/wiki/wiki/Chain-Fibers-Redux

[*Previous work is continued on the next slide.*]

2.2. Previous work [*continued*]

- Tangle [Popov, 2016] is a novel approach to consensus systems. ...
  - ...it largely abandons the idea of a heavily structured ordering and instead pushes for a directed acyclic graph of dependent transactions with later items helping canonicalise earlier items through explicit referencing.
  - For arbitrary state-changes, this dependency graph would quickly become intractable...without the hard global coherency, interaction with other systems—which tend to need an absolute degree knowledge over the system state—becomes impractical.
- Side-chains [Back et al., 2014] is a proposed addition to the Bitcoin protocol which would allow trustless interaction between the main Bitcoin chain and additional **side**-chains. ...

[*Previous work is continued on the next slide.*]

2.2. Previous work [*continued*]

- Cosmos [Kwon and Buchman, 2016] is a proposed multi-chain system in the same vein as side-chains, swapping the Nakamoto PoW consensus method for Jae Kwon's Tendermint algorithm.
  - Essentially, it describes multiple chains (operating in **zones**) each using individual instances of Tendermint, together with a means for trust-free communication via a master **hub** chain.
  - This interchain communication is limited to the transfer of digital assets ("specifically about tokens") rather than arbitrary information...
  - Validator sets for the zoned chains, and in particular the means of incentivising them, are, like side-chains, left as an unsolved problem. ...

[*Previous work is continued on the next slide.*]

2.2. Previous work [*continued*]

- ...Casper
  - is a reimagining of how a PoS consensus algorithm could be based around participants betting on which fork would ultimately become canonical. ...
  - ...is an Ethereum Foundation-centered project originally designed to be a PoS alteration to the protocol with no desire to create a fundamentally scalable blockchain. ...
  - In short, while Casper/Ethereum 2.0 and Polkadot share some fleeting similarities we believe their end goal is substantially different and that rather than competing, the two protocols are likely to ultimately co-exist under a mutually beneficial relationship for the foreseeable future.

3. Summary

- Polkadot is a **scalable heterogeneous multi-chain**.
- This means that unlike previous blockchain implementations which have focused on providing a single chain of varying degrees of generality over potential applications, Polkadot itself is designed to provide no inherent application functionality **at all**.
- Rather, Polkadot provides the bedrock "relay-chain" upon which a large number of validatable, globally-coherent dynamic data-structures may be hosted side-by-side.
- We call these data-structures "parallelised" chains or **parachains**, though there is no specific need for them to be blockchain in nature.

[*Summary is continued on the next slide.*]

3. Summary [*continued*]

- In other words, Polkadot may be considered equivalent to a set of independent chains (e.g. the set containing Ethereum, Ethereum Classic, Namecoin and Bitcoin) except for two very important points:
  - Pooled security;
  - trust-free interchain transactability.

...

3.1. The Philosophy of Polkadot

- Polkadot should provide an absolute rock-solid foundation on which to build the next wave of consensus systems, right through the risk spectrum from production-capable mature designs to nascent ideas. ...
- We see conservative, high-value chains similar to Bitcoin or Z-cash [Sasson et al., 2014] co-existing alongside lower-value "theme-chains" (such marketing, so fun) and test-nets with zero or near-zero fees.
- We see fully-encrypted, "dark", consortium chains operating alongside—and even providing services to—highly functional and open chains such as those like Ethereum.
- We see experimental new VM-based chains such as a subjective time-charged wasm chain being used as a means of outsourcing difficult compute problems from a more mature Ethereum-like chain or a more restricted Bitcoin-like chain.

[*The Philosophy of Polkadot is continued on the next slide.*]

3.1. The Philosophy of Polkadot [*continued*]

- To manage chain upgrades, Polkadot will inherently support some sort of governance structure, likely based on existing stable political systems and having a bicameral aspect similar to the Yellow Paper Council [Wood, 2016]. ...
- Whereas reparameterisation is typically trivial to arrange within a larger consensus mechanism, more qualitative changes such as replacement and augmentation would likely need to be either non-automated "soft-decrees" (e.g. through the canonicalisation of a block number and the hash of a document formally specifying the new protocol) or necessitate the core consensus mechanism to contain a sufficiently rich language to describe any aspect of itself which may need to change.
- The latter is an eventual aim, however, the former more likely to be chosen in order to facilitate a reasonable development timeline.

[*The Philosophy of Polkadot is continued on the next slide.*]

3.1. The Philosophy of Polkadot [*continued*]

- Polkadot's primary tenets and the rules within which we evaluate all design decisions are:

  Minimal Polkadot should have as little functionality as possible.

  Simple no additional complexity should be present in the base protocol than can reasonably be offloaded into middleware, placed through a parachain or introduced in a later optimisation.

  General no unnecessary requirement, constraint or limitation should be placed on parachains; Polkadot should be a test bed for consensus system development which can be optimised through making the model into which extensions fit as abstract as possible.

  Robust Polkadot should provide a fundamentally stable base-layer. In addition to economic soundness, this also means decentralising to minimise the vectors for high-reward attacks.

4. Participation in Polkadot

There are four basic roles in the upkeep of an Polkadot network: collator, fisherman, nominator and validator. ...

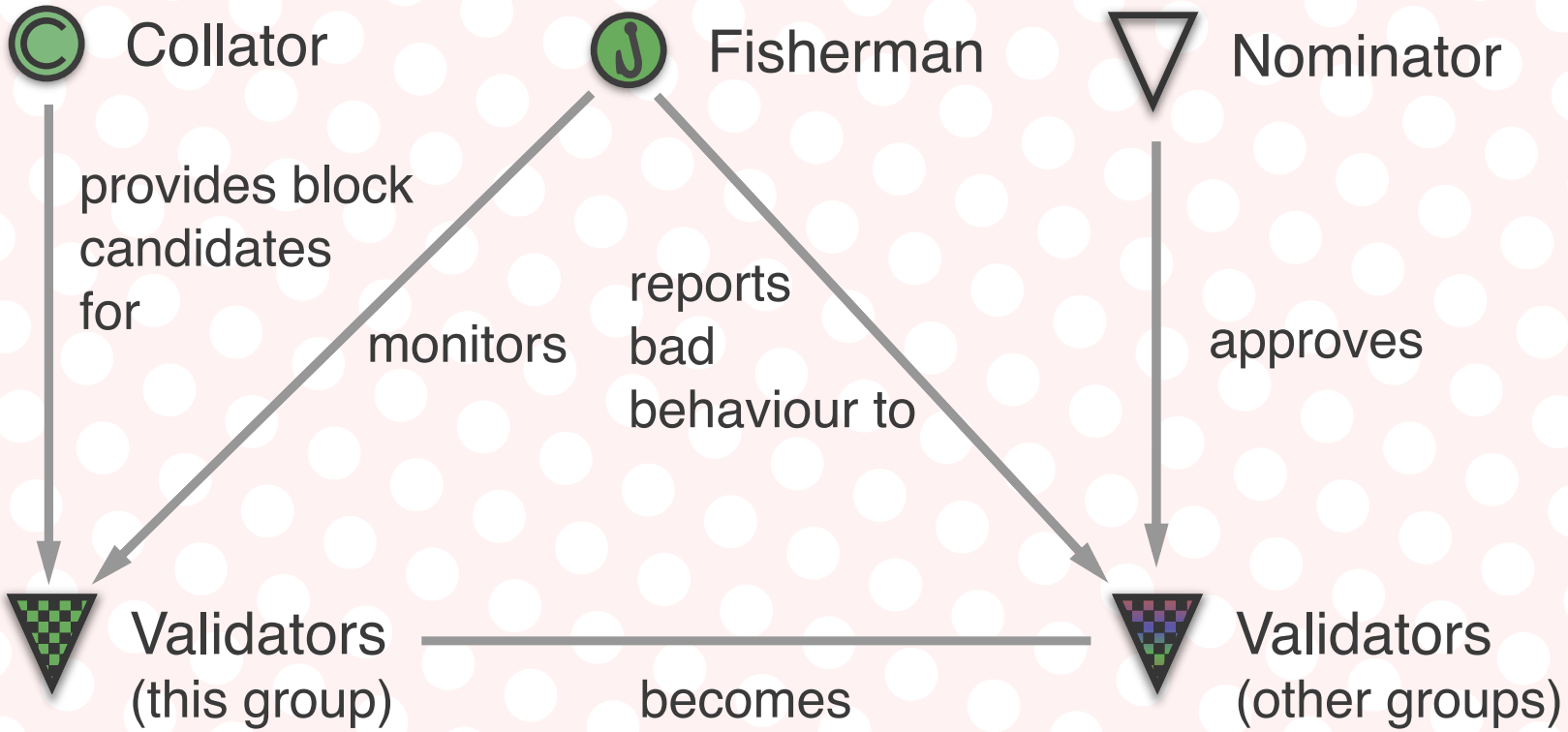

Figure: The interaction between the four roles of Polkadot.

4.1. Validators

- A validator is the highest charge and helps seal new blocks on the Polkadot network.
- The validator's role is contingent upon a sufficiently high bond being deposited, though we allow other bonded parties to nominate one or more validators to act for them and as such some portion of the validator's bond may not necessarily be owned by the validator itself but rather by these nominators.

[*Validators is continued on the next slide.*]

4.1. Validators [*continued*]

- A validator must run a relay-chain client implementation with high availability and bandwidth.
- At each block the node must be ready to accept the role of ratifying a new block on a nominated parachain.
- This process involves receiving, validating and republishing candidate blocks.
- The nomination is deterministic but virtually unpredictable much in advance.
- Since the validator cannot reasonably be expected to maintain a fully-synchronised database of all parachains, it is expected that the validator will nominate the task of devising a suggested new parachain block to a third-party, known as a collator.

[*Validators is continued on the next slide.*]

4.1. Validators [*continued*]

- Once all new parachain blocks have been properly ratified by their appointed validator subgroups, validators must then ratify the relay-chain block itself.
- This involves updating the state of the transaction queues (essentially moving data from a parachain's output queue to another parachain's input queue), processing the transactions of the ratified relay-chain transaction set and ratifying the final block, including the final parachain changes.
- A validator not fulfilling their duty to find consensus under the rules of our chosen consensus algorithm is punished. ...
- In some sense, validators are similar to the **mining pools** of current PoW blockchains. [*Emphasis added.*]

4.2. Nominators

- A nominator is a stake-holding party who contributes to the security bond of a validator.
- They have no additional role except to place risk capital and as such to signal that they trust a particular validator (or set thereof) to act responsibly in their maintenance of the network.
- They receive a pro-rata increase or reduction in their deposit according to the bond's growth to which they contribute.
- Together with collators, next, nominators are in some sense similar to the **miners** of the present-day PoW networks. [*Emphasis added.*]

4.3. Collators

- Transaction collators (collators for short) are parties who assist validators in producing valid parachain blocks.
- They maintain a "full-node" for a particular parachain; meaning that they retain all necessary information to be able to author new blocks and execute transactions in much the same way as miners do on current PoW blockchains.
- Under normal circumstances, they will collate and execute transactions to create an unsealed block, and provide it, together with a zero-knowledge proof, to one or more validators presently responsible for proposing a parachain block. ...

4.4. Fishermen

- Unlike the other two active parties, fishermen are not directly related to the block-authoring process.
- Rather they are independent "bounty hunters" motivated by a large one-off reward.
- Precisely due to the existence of fishermen, we expect events of misbehaviour to happen seldom, and when they do only due to the bonded party being careless with secret key security, rather than through malicious intent.
- The name comes from the expected frequency of reward, the minimal requirements to take part and the eventual reward size.

[*Fishermen is continued on the next slide.*]

4.4. Fishermen [*continued*]

- Fishermen get their reward through a timely proof that at least one bonded party acted illegally.
- Illegal actions include signing two blocks each with the same ratified parent or, in the case of parachains, helping ratify an invalid block. ...
- Fishermen are somewhat similar to "**full nodes**" in present-day blockchain systems that the resources needed are relatively small and the commitment of stable uptime and bandwidth is not necessary. [*Emphasis added.*] ...

5. Design Overview

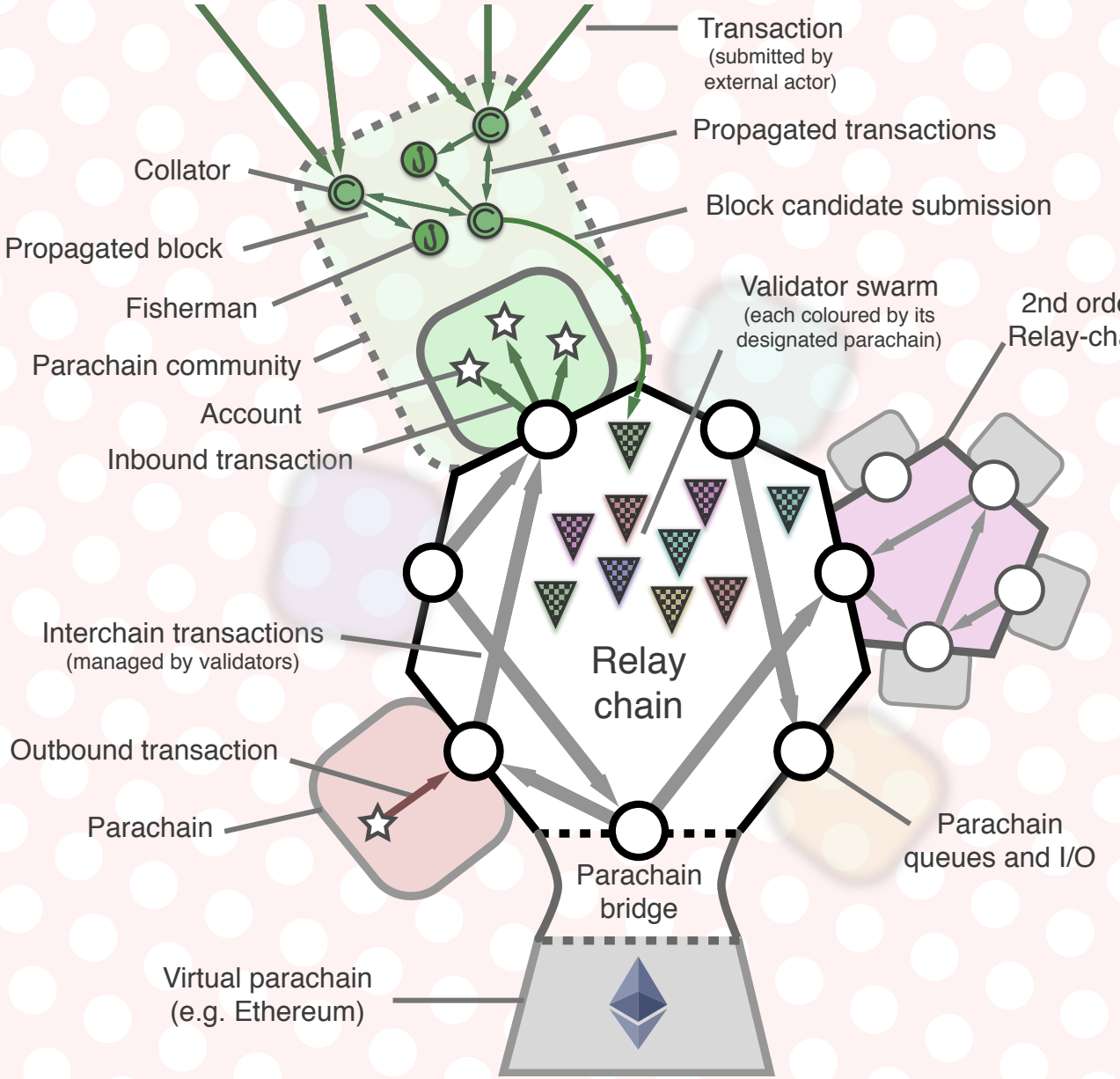

Figure: A summary schematic of the Polkadot system. ...

5.1. Consensus

- On the relay-chain, Polkadot achieves low-level consensus over a set of mutually agreed valid blocks through a modern asynchronous Byzantine fault-tolerant (BFT) algorithm.
- The algorithm will be inspired by the simple Tendermint [Kwon, 2014] and the substantially more involved HoneyBadgerBFT [Miller et al., 2016]. ...

  [*At present, it appears that Polkadot's consensus algorithm is going to be GRANDPA:*
  - *GRANDPA Block Finality in Polkadot: An Introduction (Part 1)* https://medium.com/polkadot-network/grandpa-block-finality-in-polkadot-an-introduction-part-1-d08a24a021b5
  - *Polkadot Proof-of-Concept 3: A Better Consensus Algorithm* https://medium.com/polkadot-network/polkadot-proof-of-concept-3-a-better-consensus-algorithm-e81c380a2372
  - *Byzantine Finality Gadgets* https://github.com/w3f/consensus/blob/master/pdf/grandpa.pdf

  ]

5.2. Proving the Stake

- We assume that the network will have some means of measuring how much "stake" any particular account has. ...
- We imagine validators be elected, infrequently (at most once per day but perhaps as seldom as once per quarter), through a **Nominated Proof-of-Stake** (**NPoS**) scheme. ...
- Validators are bonded heavily by their stakes; exiting validators' bonds remain in place long after the validators' duties cease (perhaps around 3 months). ...
- Misbehaviour results in punishment, such as reduction of reward or, in cases which intentionally compromise the network's integrity, the validator losing some or all of its stake to other validators, informants or the stakeholders as a whole (through burning).
- For example, a validator who attempts to ratify both branches of a fork (sometimes known as a "short-range" attack) may be identified and punished in the latter way.

[*Proving the Stake is continued on the next slide.*]

5.2. Proving the Stake [*continued*]

- Long-range "nothing-at-stake" attacks[2] are circumvented through a simple "checkpoint" latch which prevents a dangerous chain-reorganisation of more than a particular chain-depth.
- To ensure newly-syncing clients are not able to be fooled onto the wrong chain, regular "hard forks" will occur (of at most the same period of the validators' bond liquidation) that hard-code recent checkpoint block hashes into clients.
- This plays well with a further footprint-reducing measure of "finite chain length" or periodic reseting of the genesis-block.

[2] Such an attack is where the adversary forges an entirely new chain of history from the genesis block onwards. Through controlling a relatively insignificant portion of stake at the offset, they are able to incrementally increase their portion of the stake relative to all other stakeholders as they are the only active participants in their alternative history. Since no intrinsic physical limitation exists on the creation of blocks (unlike PoW where quite real computational energy must be spent), they are able to craft a chain longer than the real chain in a relatively short timespan and potentially make it the longest and best, taking over the canonical state of the network.

5.3. Parachains and Collators

- Each parachain gets similar security affordances to the relay-chain: the parachains' headers are sealed within the relay-chain block ensuring no reorganisation, or "double-spending", is possible following confirmation. ...
- Polkadot, however, also provides strong guarantees that the parachains' state transitions are valid.
- This happens through the set of validators being cryptographically randomly segmented into subsets; one subset per parachain, the subsets potentially differing per block.
- This setup generally implies that parachains' block times will be at least as long as that of the relay-chain.
- The specific means of determining the partitioning is outside the scope of this document but would likely be based either around a commit-reveal framework similar to the **RanDAO** [Qian, 2015] or use data combined from previous blocks of each parachain under a cryptographically secure hash.

[*Parachains and Collators is continued on the next slide.*]

5.3. Parachains and Collators [*continued*]

- Such subsets of validators are required to provide a parachain block candidate which is guaranteed valid (on pain of bond confiscation).
- Validity revolves around two important points;
  - firstly that it is intrinsically valid—that all state transitions were executed faithfully and that all external data referenced (i.e. transactions) is valid for inclusion.
  - Secondly, that any data which is extrinsic to its candidate, such as those external transactions, has sufficiently high availability so that participants are able to download it and execute the block manually.[3]

[3] Such a task might be shared between validators or could become the designate task of a set of heavily bonded validators known as **availability guarantors**.

[*Parachains and Collators is continued on the next slide.*]

5.3. Parachains and Collators [*continued*]

- Validators may provide only a "null" block containing no external "transactions" data, but may run the risk of getting a reduced reward if they do.
- They work alongside a parachain gossip protocol with **collators**—individuals who collate transactions into blocks and provide a non-interactive, zero-knowledge proof that the block constitutes a valid child of its parent (and taking any transaction fees for their trouble). ...

5.4. Interchain Communication

- The critical final ingredient of Polkadot is interchain communication.
- Since parachains can have some sort of information channel between them, we allow ourselves to consider Polkadot a scalable multi-chain.
- In the case of Polkadot, the communication is as simple as can be: transactions executing in a parachain are (according to the logic of that chain) able to effect the dispatch of a transaction into a second parachain or, potentially, the relay-chain.
- Like external transactions on production blockchains, they are fully asynchronous and there is no intrinsic ability for them to return any kind of information back to its origin.

[*Interchain Communication is continued on the next slide.*]

5.4. Interchain Communication [*continued*]

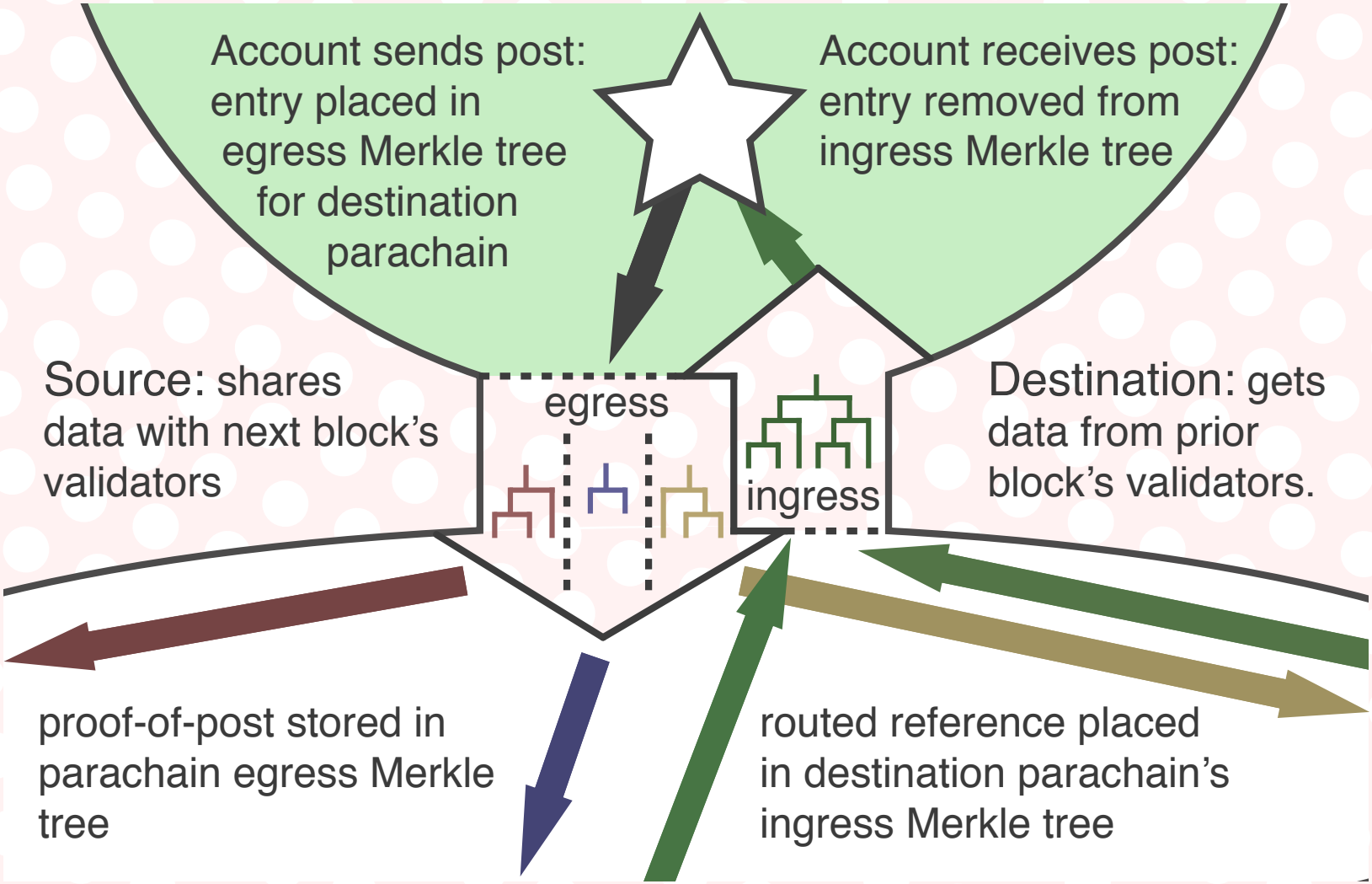

Figure: A basic schematic showing the main parts of routing for posted transactions ("posts").

5.4. Interchain Communication [*continued*]

- To ensure minimal implementation complexity, minimal risk and minimal straight-jacketing of future parachain architectures, these interchain transactions are effectively indistinguishable from standard externally-signed transactions.
- The transaction has an origin segment, providing the ability to identify a parachain, and an address which may be of arbitrary size.
- Unlike common current systems such as Bitcoin and Ethereum, interchain transactions do not come with any kind of "payment" of fee associated; any such payment must be managed through negotiation logic on the source and destination parachains.
- A system such as that proposed for Ethereum's Serenity release [Buterin, 2016b] would be a simple means of managing such a cross-chain resource payment, though we assume others may come to the fore in due course.

[*Interchain Communication is continued on the next slide.*]

5.4. Interchain Communication [*continued*]

- Interchain transactions are resolved using a simple queuing mechanism based around a Merkle tree to ensure fidelity.
- It is the task of the relay-chain maintainers to move transactions on the output queue of one parachain into the input queue of the destination parachain.
- The passed transactions get referenced on the relay-chain, however are not relay-chain transactions themselves.

[*Interchain Communication is continued on the next slide.*]

5.4. Interchain Communication [*continued*]

- To prevent a parachain from spamming another parachain with transactions, for a transaction to be sent, it is required that the destination's input queue be not too large at the time of the end of the previous block.
- If the input queue is too large after block processing, then it is considered "saturated" and no transactions may be routed to it within subsequent blocks until reduced back below the limit.
- These queues are administered on the relay-chain allowing parachains to determine each other's saturation status; this way a failed attempt to post a transaction to a stalled destination may be reported synchronously.
- (Though since no return path exists, if a secondary transaction failed for that reason, it could not be reported back to the original caller and some other means of recovery would have to take place.)

5.5. Polkadot and Ethereum

- Due to Ethereum's Turing completeness, we expect there is ample opportunity for Polkadot and Ethereum to be interoperable with each other, at least within some easily deducible security bounds.
- In short, we envision that transactions from Polkadot can be signed by validators and then fed into Ethereum where they can be interpreted and enacted by a transaction-forwarding contract.
- In the other direction, we foresee the usage of specially formatted logs (events) coming from a "break-out contract" to allow a swift verification that a particular message should be forwarded.

[*Polkadot and Ethereum is continued on the next slide.*]

5.5. Polkadot and Ethereum [*continued*]

- ...Ethereum is able to host a "break-in contract"...
- In this model, Polkadot's validator nodes would have to do little other than sign messages.
- To get the transactions actually routed onto the Ethereum network, we assume either validators themselves would also reside on the Ethereum network or, more likely, that small bounties be offered to the first actor who forwards the message on to the network (the bounty could trivially be paid to the transaction originator).

[*Polkadot and Ethereum is continued on the next slide.*]

5.5. Polkadot and Ethereum [*continued*]

- Getting transactions to be forwarded from Ethereum to Polkadot uses the simple notion of logs.
- When an Ethereum contract wishes to dispatch a transaction to a particular parachain of Polkadot, it need simply call into a special "break-out contract".
- The break-out contract would take any payment that may be required and issue a logging instruction so that its existence may be proven through a Merkle proof and an assertion that the corresponding block's header is valid and canonical.

[*Polkadot and Ethereum is continued on the next slide.*]

5.5. Polkadot and Ethereum [*continued*]

- Of the latter two conditions, validity is perhaps the most straightforward to prove.
- In principle, the only requirement is for each Polkadot node needing the proof (i.e. appointed validator nodes) to be running a fully synchronised instance of a standard Ethereum node. ...
- A more lightweight method would be to use a simple proof that the header was evaluated correctly through supplying only the part of Ethereum's state trie needed to properly execute the transactions in the block and check that the logs...are valid.
- Such "SPV-like"[4] proofs may...not be needed at all: a bond system inside Polkadot would allow bonded third-parties to submit headers at the risk of losing their bond should some other third-party (such as a "fisherman", see [*slide 61*]) provide a proof that the header is invalid...

[4] Simplified Payment Verification...a method for clients to verify transactions while keeping only a copy of all blocks headers of the longest PoW chain.

[*Polkadot and Ethereum is continued on the next slide.*]

5.5. Polkadot and Ethereum [*continued*]

- On a non-finalising PoW network like Ethereum, the canonicality is impossible to proof conclusively.
- To address this, applications that attempt to rely on any kind of chain-dependent cause-effect wait for a number of "confirmations", or until the dependent transaction is at some particular depth within the chain. ...
- So we can imagine our Polkadot-side Ethereum-interface to have some simple functions:
  - to be able to accept a new header from the Ethereum network and validate the PoW,
  - to be able to accept some proof that a particular log was emitted by the Ethereum-side break-out contract for a header of sufficient depth...and finally
  - to be able to accept proofs that a previously accepted but not-yet-enacted header contains an invalid receipt root.

[*Polkadot and Ethereum is continued on the next slide.*]

5.5. Polkadot and Ethereum [*continued*]

- To actually get the Ethereum header data itself (and any SPV proofs or validity/canonicality refutations) into the Polkadot network, an incentivisation for forwarding data is needed.
- This could be as simple as a payment (funded from fees collected on the Ethereum side) paid to anyone able to forward a useful block whose header is valid.
- Validators would be called upon to retain information relating to the last few thousand blocks in order to be able to manage forks, either through some protocol-intrinsic means or through a contract maintained on the relay chain.

5.6. Polkadot and Bitcoin

- Bitcoin interoperation presents an interesting challenge for Polkadot: a so-called "two-way peg" would be a useful piece of infrastructure to have on the side of both networks. ...
- The problem however is how the deposits can be securely controlled from a rotating validator set.
- Unlike Ethereum which is able to make arbitrary decisions based upon combinations of signatures, Bitcoin is substantially more limited, with most clients accepting only multi-signature transactions with a maximum of 3 parties. ...
- As such we believe it not unrealistic to place a reasonably secure Bitcoin interoperability "virtual parachain" between the two networks, though nonetheless a substantial effort with an uncertain timeline and quite possibly requiring the cooperation of the stakeholders within that network.

6. Protocol in Depth

The protocol can be roughly broken down into three parts:

- the consensus mechanism,
- the parachain interface and
- interchain transaction routing.

6.1. Relay-chain Operation

- The relay-chain will likely be a chain broadly similar to Ethereum in that it is state-based with the state mapping address to account information, mainly balances and (to prevent replays) a transaction counter. ...
- There will be notable differences, though:
  - Contracts cannot be deployed through transactions; following from the desire to avoid application functionality on the relay-chain, it will not support public deployment of contracts.
  - Compute resource usage ("gas") is not accounted; since the only functions available for public usage will be fixed...a flat fee will apply in all cases, allowing for more performance from any dynamic code execution that may need to be done and a simpler transaction format.
  - Special functionality is supported for listed contracts that allows for auto-execution and network-message outputs.

[*Relay-chain Operation is continued on the next slide.*]

6.1. Relay-chain Operation [*continued*]

- In the event that the relay-chain has a VM and it be based around the EVM, it would have a number of modifications to ensure maximal simplicity. ...
- If not the EVM, then a WebAssembly [WebAssembly, 2016] (wasm) back-end is the most likely alternative; in this case the overall structure would be similar, but there would be no need for the built-in contracts with Wasm being a viable target for general purpose languages rather than the immature and limited languages for the EVM.

[*Relay-chain Operation is continued on the next slide.*]

6.1. Relay-chain Operation [*continued*]

[*Polkadot will feature a form of "auto update". According to Gavin Wood (https://www.trustnodes.com/2019/01/26/polkadot-the-ethereum-2-0-or-the-mighty-race-vitalik-buterin-v-gavin-wood):*

- *Auto updates, yes. Though only within a very strict sandbox: the auto-updated software is just the core "runtime" of the blockchain. It can't do any dangerous stuff like access the filesystem or network. It can only process blocks and record stuff into the blockchain's database.*
- *We use WebAssembly for the auto-updatable core to ensure that our core runtime is consensus-safe, secure, fast, cross-platform and well-supported. As far as I know we're the only ones doing that right now. ...*
- *Those who do not explicitly hard-fork will automatically "go along with" the updates as directed by the (transparent, on-chain) governance process.*

]

6.1. Relay-chain Operation [*continued*]

- It is possible, though unlikely, that a Serenity-like "pure" chain be deployed as the relay-chain, allowing for a particular contract to manage things like the staking token balances rather than making that a fundamental part of the chain's protocol.
- At present, we feel it is unlikely this will offer a sufficiently great protocol simplification to be worth the additional complexity and uncertainty involved in developing it.

[*Relay-chain Operation is continued on the next slide.*]

6.1. Relay-chain Operation [*continued*]

- There are a number of small pieces of functionality required for administrating the consensus mechanism, validator set, validation mechanism and parachains.
- These could be implemented together under a monolithic protocol.
- However, for reasons of auguring modularity, we describe these as "contracts" of the relay-chain.
- This should be taken to mean that they are objects (in the sense of object-orientated programming) managed by the relay-chain's consensus mechanism, but not necessarily that they are defined as programs in EVM-like opcodes, nor even that they be individually addressable through the account-system.

6.2. Staking Contract

- This contract maintains the validator set.
- It manages:
  - which accounts are currently validators;
  - which are available to become validators at short notice;
  - which accounts have placed stake nominating to a validator;
  - properties of each including staking volume, acceptable payout-rates and addresses and short-term (session) identities.
- It allows an account
  - to register a desire to become a bonded validator (along with its requirements),
  - to nominate to some identity, and
  - for preexisting bonded validators to register their desire to exit this status.
- It also includes the machinery itself for the validation and canonicalisation mechanism.

6.2.1. Stake-token Liquidity

- ...if the [*staking*] token is locked in the Staking Contract under punishment of reduction, how can a substantial portion remain sufficiently liquid in order to allow price discovery? ...
- Keeping with our tenets, we elect for the simplest solution: not all tokens be staked.
- This would mean that some proportion (perhaps 20%) of tokens will forcibly remain liquid.
- Though this is imperfect from a security perspective, it is unlikely to make a fundamental difference in the security of the network; 80% of the reparations possible from bond-confiscations would still be able to be made compared to the "perfect case" of 100% staking.

[*Stake-token Liquidity is continued on the next slide.*]

6.2.1. Stake-token Liquidity [*continued*]

- The ratio between staked and liquid tokens can be targeted fairly simply through a reverse auction mechanism.
- Essentially, token holders interested in being a validator would each post an offer to the staking contract stating the minimum payout-rate that they would require to take part.
- At the beginning of each session (sessions would happen regularly, perhaps as often as once per hour) the validator slots would be filled according to each would-be validator's stake and payout rate.
  ...

6.2.2. Nominating

- It is possible to trustlessly nominate ones staking tokens to an active validator, giving them the responsibility of validators duties.
- Nominating works through an approval-voting system.
- Each would-be nominator is able to post an instruction to the staking contract expressing one or more validator identities under whose responsibility they are prepared to entrust their bond.
- Each session, nominators' bonds are dispersed to be represented by one or more validators.
- The dispersal algorithm optimises for a set of validators of equivalent total bonds.
- Nominators' bonds become under the effective responsibility of the validator and gain interest or suffer a punishment-reduction accordingly.

6.2.3. Bond Confiscation/Burning

- Certain validator behaviour results in a punitive reduction of their bond.
- If the bond is reduced below the allowable minimum, the session is prematurely ended and another started.
- A non-exhaustive list of punishable validator misbehaviour includes:
  - Being part of a parachain group unable to provide consensus over the validity of a parachain block;
  - actively signing for the validity of an invalid parachain block;
  - inability to supply egress payloads previously voted as available;
  - inactivity during the consensus process;
  - validating relay-chain blocks on competing forks.

[*Bond Confiscation/Burning is continued on the next slide.*]

6.2.3. Bond Confiscation/Burning [*continued*]

- Some cases of misbehaviour threaten the network's integrity (such as signing invalid parachain blocks and validating multiple sides of a fork) and as such result in effective exile through the total reduction of the bond.
- In other, less serious cases (e.g. inactivity in the consensus process) or cases where blame cannot be precisely allotted (being part of an ineffective group), a small portion of the bond may instead be fined. ...
- In some cases...it is necessary to enlist the support of parties external to the validation process to verify and report such misbehaviour.
- The parties get a reward for reporting such activity; their term, "fishermen" stems from the unlikeliness of such a reward. ...
- In general, it is important that the reward is sufficiently large to make verification worthwhile for the network, yet not so large...[*that*] a perverse incentive arise of misbehaving and reporting oneself for the bounty. ...

6.3. Parachain Registry

- Each parachain is defined in this registry...[*which*] includes the chain index (a simple integer), along with the validation protocol identity, a means of distinguishing between the different classes of parachain so that the correct validation algorithm can be run by validators consigned to putting forward a valid candidate. ...
- Ultimately...it may be possible to specify the validation algorithm in a way both rigorous and efficient enough that clients are able to effectively work with new parachains without a hard-fork.
- One possible avenue to this would be to specify the parachain validation algorithm in a well-established, natively-compiled, platform-neutral language such as WebAssembly. ...

[*Parachain Registry is continued on the next slide.*]

6.3. Parachain Registry [*continued*]

- **The registry is able to have parachains added only through full referendum voting**... [*Emphasis added.*]
- Additional operations include the suspension and removal of parachains.
- Suspension...is designed to be a safeguard least there be some intractable problem in a parachain's validation system.
- The most obvious instance where it might be needed is a consensus-critical difference between implementations leading validators to be unable to agree on validity or blocks. ...
- Since suspension is an emergency measure, it would be under the auspices of the dynamic validator-voting rather than a referendum.

...

[*Parachain Registry is continued on the next slide.*]

6.3. Parachain Registry [*continued*]

- The removal of parachains altogether would come only after a referendum and with which would be required a substantial grace period to allow an orderly transition to either a standalone chain or to become part of some other consensus-system.
- The grace period would likely be of the order of months and is likely to be set out on a per-chain basis in the parachain registry in order that different parachains can enjoy different grace periods according to their need.

6.4. Sealing Relay Blocks

- Sealing refers, in essence, to the process of canonicalisation; that is, a basic data transform which maps the original into something fundamentally singular and meaningful.
- Under a PoW chain, sealing is effectively a synonym for **mining**.
- In our case, it involves the collection of signed statements from validators over the validity, availability and canonicality of a particular relay-chain block and the parachain blocks that it represents.
- The mechanics of the underlying BFT consensus algorithm is out of scope for the present work. ...

  [*See on comment on slide 33 regarding GRANDPA.*]
- The proof, which takes the form of our signed statements, is placed in the relay-chain block's header together with certain other fields not least the relay-chain's state-trie root and transaction-trie root.

[*Sealing Relay Blocks is continued on the next slide.*]

6.4. Sealing Relay Blocks [*continued*]

- The sealing process takes place under a single consensus-generating mechanism addressing both the relay-chain's block and the parachains' blocks which make up part of the relay's content: parachains are not separately "committed" by their sub-groups and then collated later.
- This results in a more complex process for the relay-chain, but allows us to complete the entire system's consensus in a single stage, minimising latency and allowing for quite complex data-availability requirements which are helpful for the routing process below.
- ...Each participant (validator) has a set of information, in the form of signed-statements ("votes") from other participants, regarding each parachain block candidate as well the relay-chain block candidate.
- The set of information is two pieces of data:

[*Sealing Relay Blocks is continued on the next slide.*]

6.4. Sealing Relay Blocks [*continued*]

Availability does this validator have egress transaction-post information from this block so they are able to properly validate parachain candidates on the following block? They may vote either 1(known) or 0 (not yet known). Once they vote 1, they are committed to voting similarly for the rest of this process. Later votes that do not respect this are grounds for punishment.

Validity is the parachain block valid and is all externally-referenced data (e.g. transactions) available? This is only relevant for validators assigned to the parachain on which they are voting. They may vote either 1 (valid), -1 (invalid) or 0 (not yet known). Once they vote non-zero, they are committed to voting this way for the rest of the process. Later votes that do not respect this are grounds for punishment.

6.4. Sealing Relay Blocks [*continued*]

- All validators must submit votes; votes may be resubmitted, qualified by the rules above.
- The progression of consensus may be modelled as multiple standard BFT consensus algorithms over each parachain happening in parallel.
- Since these are potentially thwarted by a relatively small minority of malicious actors being concentrated in a single parachain group, the overall consensus exists to establish a backstop, limiting the worst-case scenario from deadlock to merely one or more void parachain blocks (and a round of punishment for those responsible).

[*Sealing Relay Blocks is continued on the next slide.*]

6.4. Sealing Relay Blocks [*continued*]

- The basic rules for validity of the individual blocks (that allow the total set of validators as a whole to come to consensus on it becoming the unique parachain candidate to be referenced from the canonical relay):
  - must have at least two thirds of its validators voting positively and none voting negatively;
  - must have over one third validators voting positively to the availability of egress queue information.
- If there is at least one positive and at least one negative vote on validity, an exceptional condition is created and the whole set of validators must vote to determine if there are malicious parties or if there is an accidental fork. ...

[*Sealing Relay Blocks is continued on the next slide.*]

6.4. Sealing Relay Blocks [*continued*]

- After all votes are counted from the full validator set, if the losing opinion has at least some small proportion (to be parameterised; at most half, perhaps significantly less) of the votes of the winning opinion, then it is assumed to be an accidental parachain fork and the parachain is automatically suspended from the consensus process.
- Otherwise, we assume it is a malicious act and punish the minority who were voting for the dissenting opinion.
- The conclusion is a set of signatures demonstrating canonicality.
- The relay-chain block may then be sealed and the process of sealing the next block begun.

6.5. Improvements for Sealing Relay Blocks

- ...While data availability within open consensus networks is essentially an unsolved problem, there are ways of mitigating the overhead placed on validator nodes. ...
- 6.5.1. **Introducing Latency** ...By requiring 33%+1 validators voting for availability only **eventually**, and not **immediately**, we can better utilise exponential data propagation and help even out peaks in data-interchange. ...
- 6.5.2. **Public Participation** ...Similar to the fishermen, there could be external parties to police the validators who claim availability. ...
- 6.5.3. **Availability Guarantors** ...Unlike normal validators, they would not switch between parachains but rather would form a single group to attest to the availability of all important interchain data. ...

6.5.4. Collator Preferences

- One important aspect of this system is to ensure that there is a healthy selection of collators creating the blocks in any given parachain. ...
- One option is to artificially weight parachain blocks in a pseudo-random mechanism in order to favour a wide variety of collators. ...
- To ensure that collators are given a reasonable fair chance of their candidate being chosen as the winning candidate in consensus, we make the specific weight of a parachain block candidate determinate on a random function connected with each collator.
- For example, taking the XOR distance measure between the collator's address and some cryptographically-secure pseudorandom number determined close to the point of the block being created (a notional "winning ticket").

[*Collator Preferences is continued on the next slide.*]

6.5.4. Collator Preferences [*continued*]

- This effectively gives each collator (or, more specifically, each collator's address) a random chance of their candidate block "winning" over all others.
- To mitigate the sybil attack of a single collator "mining" an address close to the winning ticket and thus being a favourite each block, we would add some inertia to a collator's address.
- This may be as simple as requiring them to have a baseline amount of funds in the address.
- A more elegant approach would be to weight the proximity to the winning ticket with the amount of funds parked at the address in question.
- While modelling has yet to be done, it is quite possible that this mechanism enables even very small stakeholders to contribute as a collator.

6.5.5. Overweight Blocks

- ...a validator group could reasonably form a block which takes a very long time to execute unless some particular piece of information is already known allowing a short cut, e.g. factoring a large prime.
- If a single collator knew that information, then they would have a clear advantage in getting their own candidates accepted as long as the others were busy processing the old block.
- We call these blocks **overweight**.
- Protection against validators submitting and validating these blocks largely falls under the same guise as for invalid blocks...

6.5.6. Collator Insurance

- One issue remains for validators: unlike with PoW networks, to check a collator's block for validity, they must actually execute the transactions in it.
- Malicious collators can feed invalid or overweight blocks to validators causing them **grief** (wasting their resources) and exacting a potentially substantial opportunity cost.
- To mitigate this, we propose a simple strategy on the part of validators.
- Firstly, parachain block candidates sent to validators must be signed from a relay chain account with funds; if they are not, then the validator should drop it immediately.
- Secondly, such candidates should be ordered in priority by a combination (e.g. multiplication) of the amount of funds in the account up to some cap, the number of previous blocks that the collator has successfully proposed in the past (not to mention any previous punishments), and the proximity factor to the winning ticket as discussed previously. ...

6.6. Interchain Transaction Routing

- Interchain transaction routing is one of the essential maintenance tasks of the relay-chain and its validators.
- This is the logic which governs how a posted transaction (often shortened to simply "**post**") gets from being a desired output from one **source** parachain to being a non-negotiable input of another **destination** parachain without any trust requirements.
- We choose the wording above carefully; notably we don't require there to have been a transaction in the source parachain to have explicitly sanctioned this post.
- The only constraints we place upon our model is that parachains must provide, packaged as a part of their overall block processing output, the posts which are the result of the block's execution.
- These posts are structured as several FIFO queues...

6.6.1. External Data Availability

- ...At the heart of the issue is the **availability problem** which states that since it is neither possible to make a non-interactive proof of availability nor any sort of proof of non-availability, for a BFT system to properly validate any transition whose correctness relies upon the availability of some external data, the maximum number of acceptably Byzantine nodes, plus one, of the system must attest to the data being available. ...
- Validators must contact those from the apparently offending validator sub-group who testified and either acquire and return the data to the collator or escalate the matter by testifying to the lack of availability (direct refusal to provide the data counts as a bond-confiscating offence, therefore the misbehaving validator will likely just drop the connection) and contacting additional validators to run the same test.
- In the latter case, the collator's bond is returned.
- Once a quorum of validators who can make such non-availability testimonials is reached, they are released, the misbehaving sub-group is punished, and the block reverted.

6.6.2. Posts Routing

- Each parachain header includes an egress-trie-root; this is the root of a trie containing the routing-base bins, each bin being a concatenated list of egress posts.
- Merkle proofs may be provided across parachain validators to prove that a particular parachain's block had a particular egress queue for a particular destination parachain.
- At the beginning of processing a parachain block, each other parachain's **egress** queue bound for said block is merged into our block's **ingress** queue.
- We assume strong, probably CSPR[5], sub-block ordering to achieve a deterministic operation that offers no favouritism between any parachain block pairing.
- Collators calculate the new queue and drain the egress queues according to the parachain's logic.

[5] cryptographically secure pseudo-random

[*Posts Routing is continued on the next slide.*]

6.6.2. Posts Routing [*continued*]

- The contents of the ingress queue is written explicitly into the parachain block.
- This has two main purposes: firstly, it means that the parachain can be trustlessly synchronised in isolation from the other parachains.
- Secondly, it simplifies the data logistics should the entire ingress queue not be able to be processed in a single block; validators and collators are able to process following blocks without having to source the queue's data specially.
- If the parachain's ingress queue is above a threshold amount at the end of block processing, then it is marked saturated on the relay-chain and no further messages may be delivered to it until it is cleared.
- Merkle proofs are used to demonstrate fidelity of the collator's operation in the parachain block's proof.

[*Interchain Transaction Routing is continued on the next slide.*]

6.6. Interchain Transaction Routing [*continued*]

- 6.6.3. **Critique** One minor flaw relating to this basic mechanism is the post-bomb attack...where all parachains send the maximum amount of posts possible to a particular parachain. ...
- 6.6.4. **Hyper-cube Routing** Hyper-cube routing is a mechanism which can mostly be build as an extension to the basic routing mechanism described above. Essentially, rather than growing the node connectivity with the number of parachains and sub-group nodes, we grow only with the logarithm of parachains. Posts may transit between several parachains' queues on their way to final delivery. ...
- 6.6.5. **Maximising Serendipity** ...Each block, rather than there being an unstructured repartitioning of validators among parachains, instead for each parachain sub-group, each validator would be assigned to a unique and different parachain sub-group on the following block. ...

6.7. Parachain Validation

- A validator's main purpose is to testify, as a well-bonded actor, that a parachain's block is valid, including but not limited to any state transition, any external transactions included, the execution of any waiting posts in the ingress queue and the final state of the egress queue. ...
- Once the validator sealed the previous block they are free to begin working to provide a candidate parachain block candidate for the next round of consensus.
- Initially, the validator finds a parachain block candidate through a **parachain collator** (described next) or one of its co-validators.
- The parachain block candidate data includes the block's header, the previous block's header, any external input data included..., egress queue data and internal data to prove state-transition validity...

[*Parachain Validation is continued on the next slide.*]

6.7. Parachain Validation [*continued*]

- Simultaneously, if not yet done, the validator will be attempting to retrieve information pertaining to the previous block's transition, initially from the previous block's validators and later from all validators signing for the availability of the data.
- Once the validator has received such a candidate block, they then validate it locally.
- The validation process is contained within the parachain class's validator module, a consensus-sensitive software module that must be written for any implementation of Polkadot...
- The process takes the previous block's header and verifies its identity through the recently agreed relay-chain block in which its hash should be recorded.
- Once the parent header's validity is ascertained, the specific parachain class's validation function may be called. ...

[*Parachain Validation is continued on the next slide.*]

6.7. Parachain Validation [*continued*]

- Most such validation functions will first check the header-fields which are able to be derived directly from the parent block (e.g. `parent_hash`, `number`).
- Following this, they will populate any internal data structures as necessary in order to process transactions and/or posts. ...
- Once done, the ingress posts and external transactions (or whatever the external data represents) will be enacted, balanced according to chain's specification. ...
- Through this enactment, a series of egress posts will be created and it will be verified that these do indeed match the collator's candidate.
- Finally, the properly populated header will be checked against the candidate's header.
- With a fully validated candidate block, the validator can then vote for the hash of its header and send all requisite validation information to the co-validators in its sub-group.

6.7.1. Parachain Collators

- Parachain collators are unbonded operators who fulfill much of the task of miners on the present-day blockchain networks.
- They are specific to a particular parachain.
- In order to operate they must maintain both the relay-chain and the fully synchronised parachain.
- The precise meaning of "fully synchronised" will depend on the class of parachain, though will always include the present state of the parachain's ingress queue.
- In Ethereum's case it also involves at least maintaining a Merkle-tree database of the last few blocks, but might also include various other data structures including Bloom filters for account existence, familial information, logging outputs and reverse lookup tables for block number.

[*Parachain Collators is continued on the next slide.*]

6.7.1. Parachain Collators [*continued*]

- In addition to keeping the two chains synchronised, it must also "fish" for transactions by maintaining a transaction queue and accepting properly validated transactions from the public network.
- With the queue and chain, it is able to create new candidate blocks for the validators chosen at each block (whose identity is known since the relay-chain is synchronised) and submit them, together with the various ancillary information such as proof-of-validity, via the peer network.
- For its trouble, it collects all fees relating to the transactions it includes. ...

6.8. Networking

- Networking on traditional blockchains like Ethereum and Bitcoin has rather simple requirements.
- All transactions and blocks are broadcast in a simple undirected gossip.
- Synchronisation is more involved, especially with Ethereum but in reality this logic was contained in the peer strategy rather than the protocol itself which resolved around a few request and answer message types. ...
- The requirements for Polkadot are rather more substantial.
- Rather then a wholly uniform network, Polkadot has several types of participants each with different requirements over their peer makeup and several network "avenues" whose participants will tend to converse about particular data. ...
- To ensure an efficient transport mechanism, a "flat" overlay network—like Ethereum's devp2p—where each node does not (non-arbitrarily) differentiate fitness of its peers is unlikely to be suitable. ...

6.8.1. The Problem of Peer Churn

- In the basic protocol proposal, each of these subsets constantly alter randomly with each block as the validators assigned to verify the parachain transitions are randomly elected.
- This can be a problem should disparate (non-peer) nodes need to pass data between each other. ...
- ...a reasonable optimisation to help minimise latency would be to restrict the volatility of these parachain validator sets, either reassigning the membership only between series of blocks...or by rotating membership in an incremental fashion, e.g. changing by one member at a time...
- By limiting the amount of peer churn, and ensuring that advantageous peer connections are made well in advance through the partial predictability of parachain sets, we can help ensure each node keep a permanently serendipitous selection of peers.

6.8.2. Path to an Effective Network Protocol

- Likely the most effective and reasonable development effort will focus on utilising a pre-existing protocol rather than rolling our own.
- Several peer-to-peer base protocols exist that we may use or augment including Ethereum's own devp2p [Wood, 2014a], IPFS's libp2p [lib,] and GNU's GNUnet [Bennett et al., 2002]. ...

7. Practicalities of the Protocol

[*This slide is intentionally left blank.*]

7.1. Interchain Transaction Payment

- While a great amount of freedom and simplicity is gained through dropping the need for a holistic computation resource accounting framework like Ethereum's **gas**, this does raise an important question: without gas, how does one parachain avoid another parachain from forcing it to do computation?
- ...we can imagine a "break-in" contract within a parachain which allows a validator to be guaranteed payment in exchange for the provision of a particular volume of processing resources. ...
- ...we can imagine a secondary "break-out" contract in the source chain.
- The two contracts together would form a bridge, recognising each other and providing value-equivalence. ...
- Calling into another such chain would mean proxying through this bridge, which would provide the means of negotiating the value transfer between chains in order to pay for the computation resources required on the destination parachain.

7.2. Additional Chains

- While the addition of a parachain is a relatively cheap operation, it is not free.
- More parachains means fewer validators per parachain and, eventually, a larger number of validators each with a reduced average bond.
- While the issue of a smaller coercion cost for attacking a parachain is mitigated through fishermen, the growing validator set essentially forces a higher degree of latency due to the mechanics of the underlying consensus method.
- Furthermore each parachain brings with it the potential to grief validators with an over-burdensome validation algorithm.
- As such, there will be some "price" that validators and/or the stake-holding community will extract for the addition of a new parachain.

[*Additional Chains is continued on the next slide.*]

7.2. Additional Chains [*continued*]

- This market for chains will possibly see the addition of either:
  - Chains that likely have zero net contribution paying (in terms of locking up or burning staking tokens) to be made a part (e.g. consortium chains, Doge-chains, app-specific chains);
  - chains that deliver intrinsic value to the network through adding particular functionality difficult to get elsewhere (e.g. confidentiality, internal scalability, service tie-ins).
- Essentially, the community of stakeholders will need to be incentivized to add child chains—either financially or through the desire to add featureful chains to the relay.
- It is envisioned that new chains added will have a very short notice period for removal, allowing for new chains to be experimented with without any risk of compromising the medium or long-term value proposition.

8. Conclusion

- We have outlined a direction one may take to author a scalable, heterogeneous multi-chain protocol with the potential to be backwards compatible to certain, pre-existing blockchain networks.
- Under such a protocol, participants work in enlightened self-interest to create an overall system which can be extended in an exceptionally free manner and without the typical cost for existing users that comes from a standard blockchain design.
- We have given a rough outline of the architecture it would take including the nature of the participants, their economic incentives and the processes under which they must engage.
- We have identified a basic design and discussed its strengths and limitations; accordingly we have further directions which may ease those limitations and yield further ground towards a fully scalable blockchain solution.

8.1. Missing Material and Open Questions

- Network forking is always a possibility from divergent implementations of the protocol.
- The recovery from such an exceptional condition was not discussed.
- Given the network will necessarily have a non-zero period of finalisation, it should not be a large issue to recover from the relay-chain forking, however will require careful integration into the consensus protocol.

[*Missing Material and Open Questions is continued on the next slide.*]

8.1. Missing Material and Open Questions [*continued*]

- Bond-confiscation and conversely reward provision has not been deeply explored.
- At present we assume rewards are provided under a winner-takes-all basis: this may not give the best incentivisation model for fishermen.
- A short-period commit-reveal process would allow many fishermen to claim the prize giving a fairer distribution of rewards, however the process could lead to additional latency in the discovery of misbehaviour.

References


The specs for libp2p and associated submodules.
https://github.com/libp2p/specs.


Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timon, J., and Wuille, P. (2014).
Enabling blockchain innovations with pegged sidechains.


Bennett, K., Grothoff, C., Horozov, T., Patrascu, I., and Stef, T. (2002).
Gnunet-a truly anonymous networking infrastructure.
In *In: Proc. Privacy Enhancing Technologies Workshop (PET*. Citeseer.


Buterin, V. (2013).
Ethereum: A next-generation smart contract and decentralized application platform.
https://github.com/ethereum/wiki/wiki/White-Paper.


Buterin, V. (2016a).
Ethereum 2.0 mauve paper.


Buterin, V. (2016b).
Serenity poc2.

[*References is continued on the next slide.*]

References [*continued*]


Copeland, C. and Zhong, H. (2016).
Tangaroa: a byzantine fault tolerant raft.
http://www.scs.stanford.edu/14au-cs244b/labs/projects/copeland_zhong.pdf.


Kwon, J. (2014).
Tendermint: Consensus without mining.
http://tendermint.com/docs/tendermint.pdf.


Kwon, J. and Buchman, E. (2016).
Cosmos: A network of distributed ledgers.
https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md.


Larimer, D. (2013).
Bitshares.
http://docs.bitshares.org/bitshares/history.html.


Maymounkov, P. and Mazières, D. (2002).
Kademlia: A peer-to-peer information system based on the xor metric.
In *IPTPS '01 Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65.

[*References is continued on the next slide.*]

References [*continued*]


Miller, A., Xia, Y., Croman, K., Shi, E., and Song, D. (2016).
The honey badger of bft protocols.
Technical report, Cryptology ePrint Archive 2016/199.


Nakamoto, S. (2008).
Bitcoin: A peer-to-peer electronic cash system.
https://bitcoin.org/bitcoin.pdf.


Nxt community (2013).
Whitepaper: Nxt.
http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt.


Ongaro, D. and Ousterhout, J. (2014).
In search of an understandable consensus algorithm.
In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319.


Parity (2016).
Parity ethereum client.
https://parity.io.

[*References is continued on the next slide.*]

References [*continued*]


Popov, S. (2016).
The tangle.
https://www.iotatoken.com/IOTA_Whitepaper.pdf.


Qian, Y. (2015).
Randao.
https://github.com/randao/randao.


Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. (2014).
Zerocash: Decentralized anonymous payments from bitcoin.
In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE.


Snow, P., Deery, B., Lu, J., Johnston, D., and Kirb, P. (2014).
Factom: Business processes secured by immutable audit trails on the blockchain.
https://raw.githubusercontent.com/FactomProject/FactomDocs/master/Factom_Whitepaper.pdf.


WebAssembly (2016).
http://webassembly.org/.

[*References is continued on the next slide.*]

References [*continued*]


Wood, G. (2014a).
Devp2p wire protocol.
https://github.com/ethereum/wiki/wiki/libp2p-Whitepaper.


Wood, G. (2014b).
Ethereum: a secure decentralised generalised transaction ledger.
http://gavwood.com/paper.pdf.


Wood, G. (2016).
Yellow paper committee.
https://github.com/gavofyork/curly-engine.

Appendices

Appendix A: Functional Components

- Seen from a high-level, the Parity Polkadot Platform stack has a number of functional components and any development of it will require a substantial amount of research and development prior to it being completed.
- Some components are dependent on others existing, others are independent.
- Some are very important for the platform to function properly, others are **nice-to-have**s.
- Some are of indeterminate complexity and have not yet been deemed feasible, others are relatively straight-forward.
- Here we'll try to list as many such components as possible along with where they would fit into our development roadmap.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Networking subsystem This is the means by which a peer network is formed and maintained. Simple alterations of existing peer-to-peer networking libraries (devp2p most likely) will be sufficient for an initial system. However, additional research and development will be necessary to ensure that as the network grows, the network topology becomes increasingly structured allowing for optimal data logistics. For the final deployment, adaptations of libp2p, devp2p and GNUnet should be first considered. If requirements are not likely to be met then a new protocol should be considered.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Consensus mechanism Proof-of-authority consensus mechanism supporting rich validator statements and allowing multiple independent items to be agreed upon under a single series based upon subjective reception of the partial set of validator statements. The mechanism should allow the proof of misbehaviour for the dismissal of malicious validators but need not involve any staking mechanism. A substantial amount of research and prototyping will precede the development of this component.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Proof-of-stake chain Extending the consensus mechanism into proof-of-stake territory; this module includes staking tokens, managing entry and exit from the validator pool, a market mechanism for determining validator rewards, finalising the approval-voting nomination mechanism and managing bond-confiscation and dismissal. It includes a substantial amount of research and prototyping prior to final development.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Parachain implementation A first parachain implementation, likely to be based heavily on an existing blockchain protocol such as Bitcoin or (more likely, since it provides for rich transactions) Ethereum. This will include an integration with the proof-of-stake chain, allowing the parachain to gain consensus without its own internal consensus mechanism.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Transaction processing subsystem An evolution of the parachain and relay-chain, this will allow for transactions to be sent, received and propagated. It includes the designs of transaction queuing and optimised transaction routing on the network layer.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Transaction-routing subsystem This introduces more specifics into the relay-chain's behaviour. Initially, adding transactability into parachains will be needed. Following that, with two parachains hard-coded into the relay-chain, it will include the management of the ingress/egress queues. Eventually this will develop along with the network protocol into a means of directed transaction propagation, ensuring independent parachain collators are not overly exposed to transactions that are not of interest.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Relay chain This is the final stage of the relay-chain, allowing the dynamic addition, removal and emergency pausing of parachains, the reporting of bad behaviour and includes implementation of the "fisherman" functionality.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Independent collators This is the delivery of an alternative chain-specific collator functionality. It includes proof creation (for collators), parachain misbehaviour detection (for fishermen) and the validation function (for validators). It also includes any additional networking required to allow the two to discover and communicate.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Network dynamics modelling and research The overall dynamics of this protocol should be researched thoroughly. This can happen both through offline network modelling and through empirical evidence through simulated nodes. The latter is dependent on the relay-chain and will include the development of a structured logging protocol allowing nodes to submit detailed reports on their actions to a central hub for collation.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Network intelligence As a complex decentralised multi-party network, a network intelligence hub similar to `http://ethstats.net` is needed to monitor the life-signs of the network as a whole and flag potentially disruptive behaviour. Structured logging should be used to generate and visualise this data in real-time for maximum efficiency. It is dependent on the relay-chain being in a reasonably complete state.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Information publication platform This is a mechanism for publishing data relating to the blockchain, and effectively means a decentralised content-discovery network. Initially it can be handled by basic peer-to-peer lookups but for deployment will likely require a more structured and robust solution since availability will become a critical piece of information. IPFS integration may be the most sensible means of achieving these goals.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Javascript interaction bindings The main means of interacting with the network will likely follow the example of Ethereum and as such high-quality Javascript bindings are critical to have. Our bindings will cover interactions with the relay-chain and the initial parachain and as such depends on them.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Governance Initially, this will be a meta-protocol on the relay-chain for managing exceptional events such as hard-forks, soft-forks and protocol reparameterisation. It will include a modern structure to help manage conflict and prevent live-lock. Ultimately, this may become a full meta-protocol layer able to enact changes normally reserved for hard-forks. Requires the relay chain.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Interaction platform A platform by which "typical" users are able to interact with the system, along with minimal functionality to facilitate common tasks such as entering the bond process, voting, nomination, becoming a validator, fisherman or collator and staking token transfer. Depends upon having a functional relay-chain.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Light clients Light-client technology for both the relay-chain and any parachains developed. This will allow clients to be able to gain information regarding activity on the chains with a high degree of trust-freedom and without any substantial requirement of storage or bandwidth. Depends upon the relay-chain.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Parachain UI A multi-chain, multi-token wallet and identity management system. It requires light-client technology and the interaction platform. This is not needed for any initial network deployment.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

On-chain Dapp services There may be various services that will need to be deployed on any initial parachains such as registration hubs for APIs, names, natural-language specifications and code. This depends on the parachain but is not strictly needed for any initial network deployment.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Application development tools This includes various bits of software required to help developers. Examples include compilers, key management tools, data archivers and VM simulators. Many others exist. These will need to be developed as required. Technologies will be chosen partly to minimise the need for such tools. Many will not be strictly required.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Ethereum-as-a-parachain Trust-free bridge to Ethereum and back again, allowing posted transactions to be routed from parachains into Ethereum (and treated like any other transaction of external origin) and allow Ethereum contracts to be able to post transactions to parachains and the accounts therein. Initially, this will be modelled to ascertain feasibility and get any structural limitations based around the number of validators required by the consensus process, a component on which it is dependent. A proof-of-concept can be built following this and final development will be dependent on the relay-chain itself.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Bitcoin-RPC compatibility layer A simple RPC compatibility layer for the relay-chain allowing infrastructure already built using that protocol to be interoperable with Polkadot and as such minimise effort for support. A stretch goal requiring the relay-chain.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Web 2.0 bindings Bindings into common Web 2.0 technology stacks to facilitate the usage of Polkadot infrastructure with legacy systems. A stretch goal dependent on the initial parachains and any on-chain infrastructure to be exposed.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

zk-SNARK parachain example A parachain utilising the properties of zk-SNARKs in order to ensure identities of transactors on it are kept private. A stretch goal dependent on the relay-chain.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Encrypted parachain example A parachain that keeps each item of state encrypted and signed. These ensure the enforcement of access controls over inspection and modification of the data therein allowing commercial regulated operations to conform as necessary. Would include proof-of-authority mechanisms to allow Polkadot validators to guarantee some degree of correctness of state transitions without gaining exposure to the underlying information. A stretch goal depending on the relay-chain.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Trust-free Bitcoin bridge Trust-free Bitcoin "two-way-peg" functionality. This would possibly use threshold signatures or alternatively an n of m multi-signature Bitcoin account together with SPV proofs & specialised fishermen. Development is predicated on an initial feasibility analysis giving a favourable result. Some additional functionality may need to be added to (or unlocked from) the Bitcoin protocol to support this functionality.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Abstract/low-level decentralised applications Trust-free token exchange, asset-tracking infrastructure, crowd-sale infrastructure.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

Contract language Certainly not an integral part of the project, but a nice stretch-goal nonetheless. Would include a safe contract language together with tutorials, guidelines and educational tools. It may include means of making formal proofs as per the original Solidity language vision or could be based around some other programming paradigm which helps minimise critical process errors such as functional programming or condition-orientated programming.

[*Functional Components is continued on the next slide.*]

Appendix A: Functional Components [*continued*]

IDE Based on the contract language, this would facilitate editing, collaboration, publication and debugging of contracts on the parachains. A far stretch goal.

Appendix B: Frequently Asked Questions

Is Polkadot designed to replace (insert blockchain here) No.

The goal of Polkadot is to provide a framework under which new blockchains may be created and to which existing blockchains can, if their communities desire, be transitioned.

[*Frequently Asked Questions is continued on the next slide.*]

Appendix B: Frequently Asked Questions [*continued*]

Is Polkadot designed to replace (**insert crypto-currency here**)

No. Polkadot tokens are neither intended nor designed to be used as a currency. They would make a bad currency: most will remain illiquid in the staking system and those that are liquid will face substantial fees for transfer of ownership. Rather, the purpose of Polkadot tokens is to be a direct representation of stake in the Polkadot network.

[*Frequently Asked Questions is continued on the next slide.*]

Appendix B: Frequently Asked Questions [*continued*]

What is the inflation rate for Polkadot staking tokens The Polkadot staking token base expansion is unlimited. It rises and lowers according to market effects in order to target a particular proportion of tokens held under long-term bond in the validation process.

[*Frequently Asked Questions is continued on the next slide.*]

Appendix B: Frequently Asked Questions [*continued*]

Why does staking token ownership reflect stakeholding This is a mechanism realised by the fact that they underpin the network's security. As such their value is tied to the overall economic value that Polkadot provides. Any actors who gain overall value from Polkadot operating correctly are incentivised to ensure it continues to do so. The best means of doing so is to take part in the validation process. This generally implies ownership of staking tokens.

[*Supplementary Material*]

[*Supplementary Material*]

- Polkadot Telemetry
  https://telemetry.polkadot.io/#/Alexander
- Polkadot Apps Portal
  https://poc-3.polkadot.io/#/staking
- Instructions for becoming a validator
  https://github.com/paritytech/polkadot/wiki/Validating-on-PoC-3-%22Alexander%22
- Where to ask for DOTs
  https://riot.im/app/#/room/#polkadot-watercooler:matrix.org
- Sam's public key
  `5HNYsM8ijjB46xFapowp4PrUtR9NPEP8NQsBx8iRrqMDBox1`