Aleph: Efficient Atomic Broadcast in Asynchronous Networks with Byzantine Nodes

Szymon Żak

Institute of Theoretical Computer Science, Jagiellonian University

January 7, 2021

Problem

- Why is a consensus protocol necessary?

Problem

- Why is a consensus protocol necessary?
- How to reach consensus in system, when some of the components fail?

Problem

- Why is a consensus protocol necessary?
- How to reach consensus in system, when some of the components fail?
- How many components may fail?

Assumptions

- Point to point communication between nodes
- Guaranteed delivery

Goals of consensus

- **Safety**: Every two honest nodes should make the same decision.

Goals of consensus

- **Safety**: Every two honest nodes should make the same decision.
- **Liveness**: Every honest node eventually makes a decision.

Goals of consensus

- **Safety**: Every two honest nodes should make the same decision.
- **Liveness**: Every honest node eventually makes a decision.
- **Non-triviality**: If all honest nodes start with $b = 1$, then all honest nodes should decide $d = 1$.

Goals of consensus

- **Safety**: Every two honest nodes should make the same decision.
- **Liveness**: Every honest node eventually makes a decision.
- **Non-triviality**: If all honest nodes start with $b = 1$, then all honest nodes should decide $d = 1$.

A consensus protocol is called **Byzantine Fault-Tolerant** if it achieves safety and liveness in the presence of Byzantine faults (dishonest nodes).

Types of protocols

Synchronous

There exists a (known) bound Δ such that each message sent at time t is delivered before time $t + \Delta$.

Types of protocols

Synchronous

There exists a (known) bound Δ such that each message sent at time t is delivered before time $t + \Delta$.

Partially synchronous

There exists a known bound Δ, and an unknown point in time GST (Global Stabilization Time), such that each message sent after GST reaches its destination at most Δ time later. Thus, after GST the communication becomes synchronous with delay Δ.

Types of protocols

Synchronous

There exists a (known) bound Δ such that each message sent at time t is delivered before time $t + \Delta$.

Partially synchronous

There exists a known bound Δ, and an unknown point in time GST (Global Stabilization Time), such that each message sent after GST reaches its destination at most Δ time later. Thus, after GST the communication becomes synchronous with delay Δ.

Asynchronous

No assumptions: every message gets delivered but the delay can be arbitrarily large.

Partially synchronous protocols

Theorem (Castro Liskov, 1999)

There exists a BFT binary consensus protocol that is resilient to $\sim \frac{1}{3}$ faults in the partially synchronous network model.

Partially synchronous protocols

Theorem (Castro Liskov, 1999)

There exists a BFT binary consensus protocol that is resilient to $\sim \frac{1}{3}$ faults in the partially synchronous network model.

Such protocols were known already in 80's (e.g. Streamlet, PBFT, HotStuff...).

Asynchronous protocols

Theorem (Fischer, Lynch Patterson, 1985)

It is impossible to have a deterministic protocol that solves consensus in an asynchronous system in which at least one process may fail by crashing.

Asynchronous protocols

Theorem (Fischer, Lynch Patterson, 1985)

It is impossible to have a deterministic protocol that solves consensus in an asynchronous system in which at least one process may fail by crashing.

A standard technique of circumventing FLP theorem is to give up determinism by introducing randomization in a form of coin tossing.

Asynchronous protocols

Theorem (Fischer, Lynch Patterson, 1985)

It is impossible to have a deterministic protocol that solves consensus in an asynchronous system in which at least one process may fail by crashing.

A standard technique of circumventing FLP theorem is to give up determinism by introducing randomization in a form of coin tossing.

We have two classes of such randomized algorithms:

- all processes have access to one shared coin
- each process uses its own local coin

Aleph Protocol

The main idea of algorithm is construction of a partially ordered set and the algorithm for reaching a consensus on its extension to a total order.

Setup

- Network of $3f + 1$ nodes
- Adversary controls f nodes and network delays
- We want to make *iterated consensus*

Aleph Protocol

Unit

The basic building block of the protocol is a unit, a data structure created by a single node and propagated throughout the network. Units are used as containers for messages, and the protocol uses them to guarantee the consistency of the system. They are sent over nodes by RBC (Reliable Broadcast) protocol.

Aleph Protocol

Unit

The basic building block of the protocol is a unit, a data structure created by a single node and propagated throughout the network. Units are used as containers for messages, and the protocol uses them to guarantee the consistency of the system. They are sent over nodes by RBC (Reliable Broadcast) protocol.

Definition

Every unit has the following attributes:

- creator - index and signature of unit's creator.
- parents - hashes of previous units.
- data - additional data to be included in the unit.

Communication history DAGs

DAGs

To define the basic rules of creating units note first that this DAG structure induces a partial ordering on the set of units. To emphasize this fact, we often write $V \leqslant U$ if either V is a parent of U, or more generally (transitive closure) that V can be reached from U by taking the "parent pointer" several times. We denote the DAG-round of a unit U by $R(U)$.

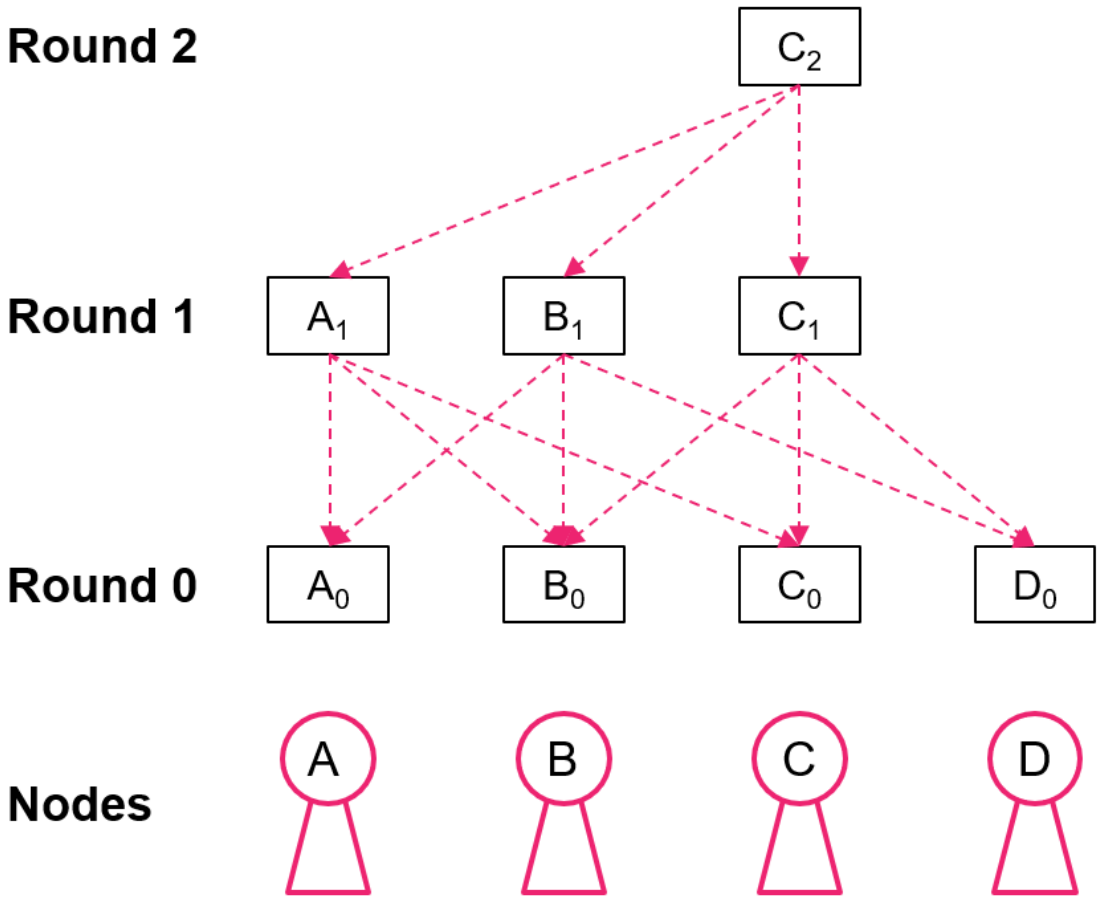
Communication history DAG

Definition

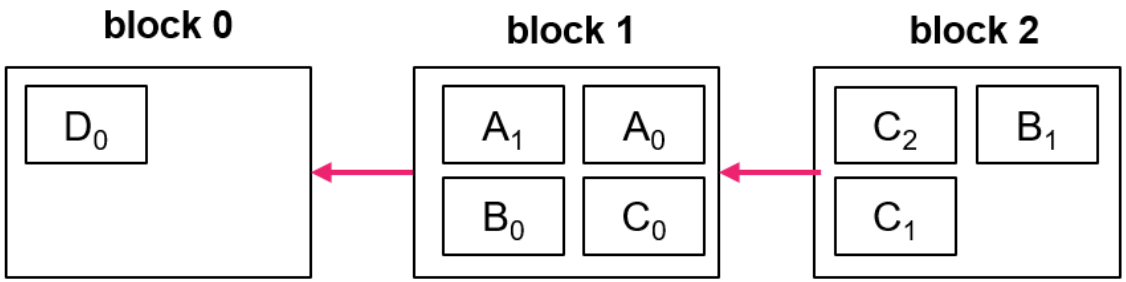
We say that a set of units D created by $N = 3f + 1$ nodes forms a communication history DAG if the parents of every unit in D also belong to D and additionally the following conditions hold true:

- **Chains** For each honest node $P_i \in P$, the set of units in D created by P_i forms a chain.
- **Dissemination** Every $round - r$ unit in D has at least $2f + 1$ parents of round $r1$.
- **Diversity** Every unit in D has parents created by pairwise distinct nodes.

Building DAGs

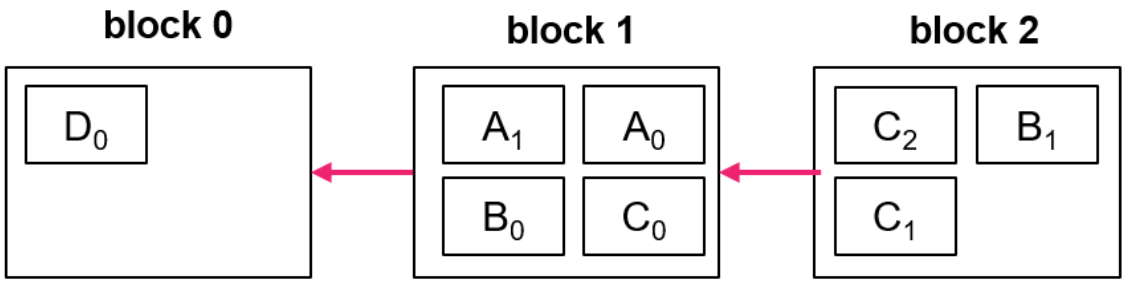


Aleph - general idea



- Choose one unit from every level as a ‘head’.
- i-th block consists of transaction from units below that are below i-th head (including it) but not below any previous head.

Aleph - general idea



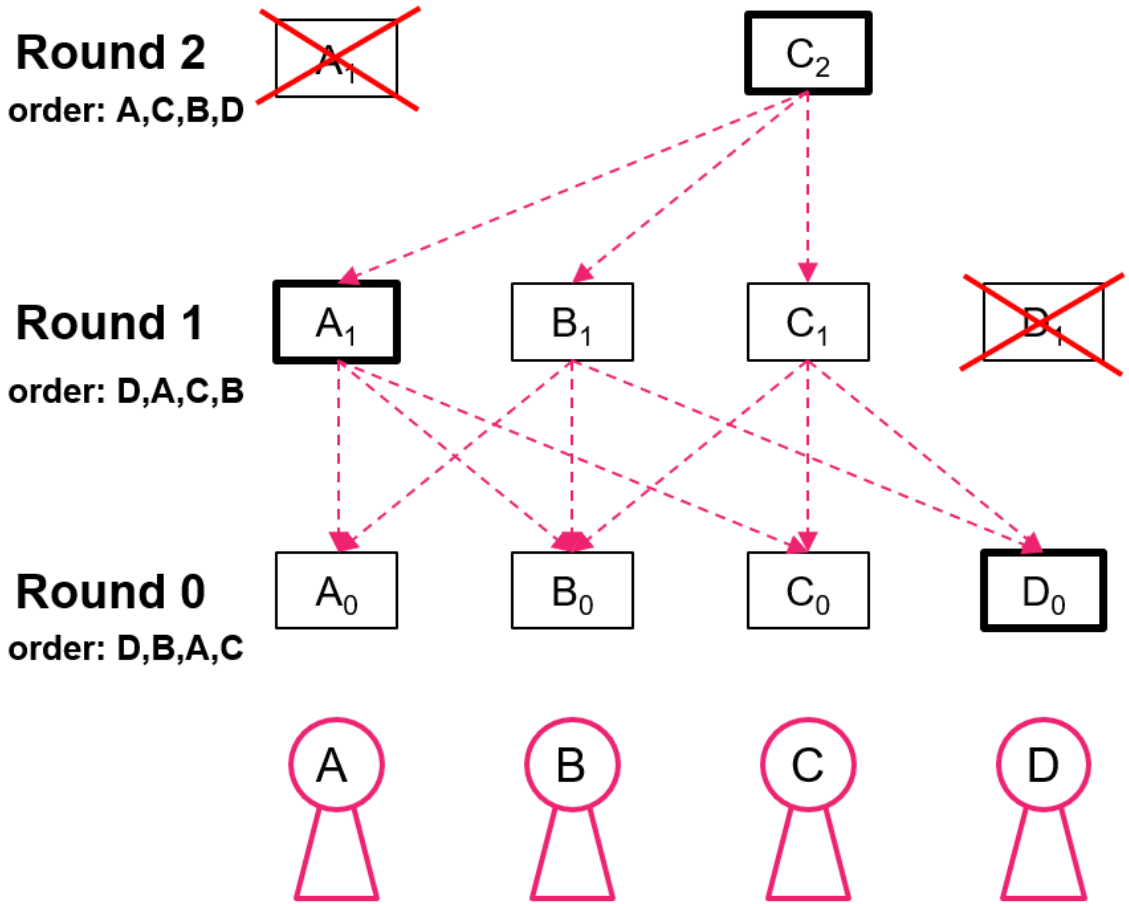
- Choose one unit from every level as a 'head'.
- i-th block consists of transaction from units below that are below i-th head (including it) but not below any previous head.

Problem

How do we know that nodes agree on which unit to choose for head?

Aleph - general idea

1. Define an order of nodes for every round.
2. Make a consensus on availability of every unit.
3. The first available unit according to order will be chosen as head.



Consensus on availability

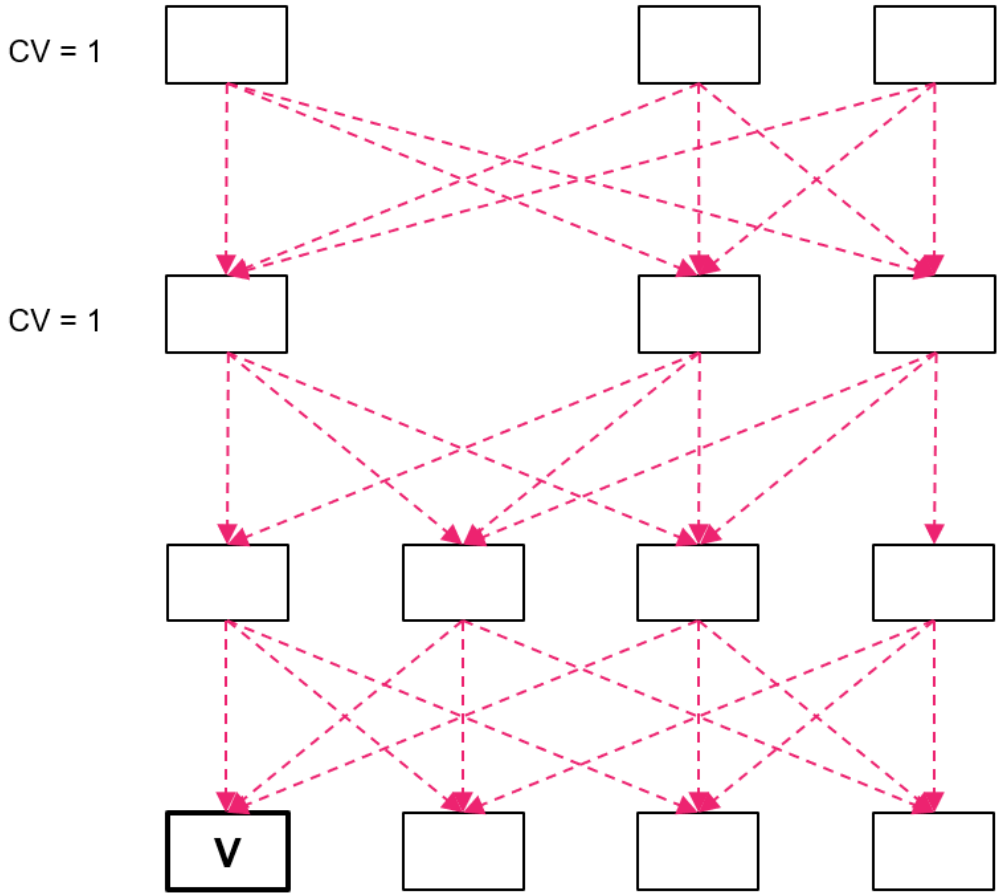
$$\text{Vote}_U(V) = \begin{cases} [\text{U¿V}] & \text{if R(U)} \leqslant \text{R(V)+1} \\ i & \text{if all children of U vote i} \\ \mathit{CommonVote}(R(U), R(V)) & \text{otherwise} \end{cases}$$

$$\text{CommonVote(i,j)} = \begin{cases} 1 & \text{if i ¡ j+4} \\ 0 & \text{if i} = \text{j} + 4 \\ \mathit{CommonRandomness}(j) & \text{otherwise} \end{cases}$$

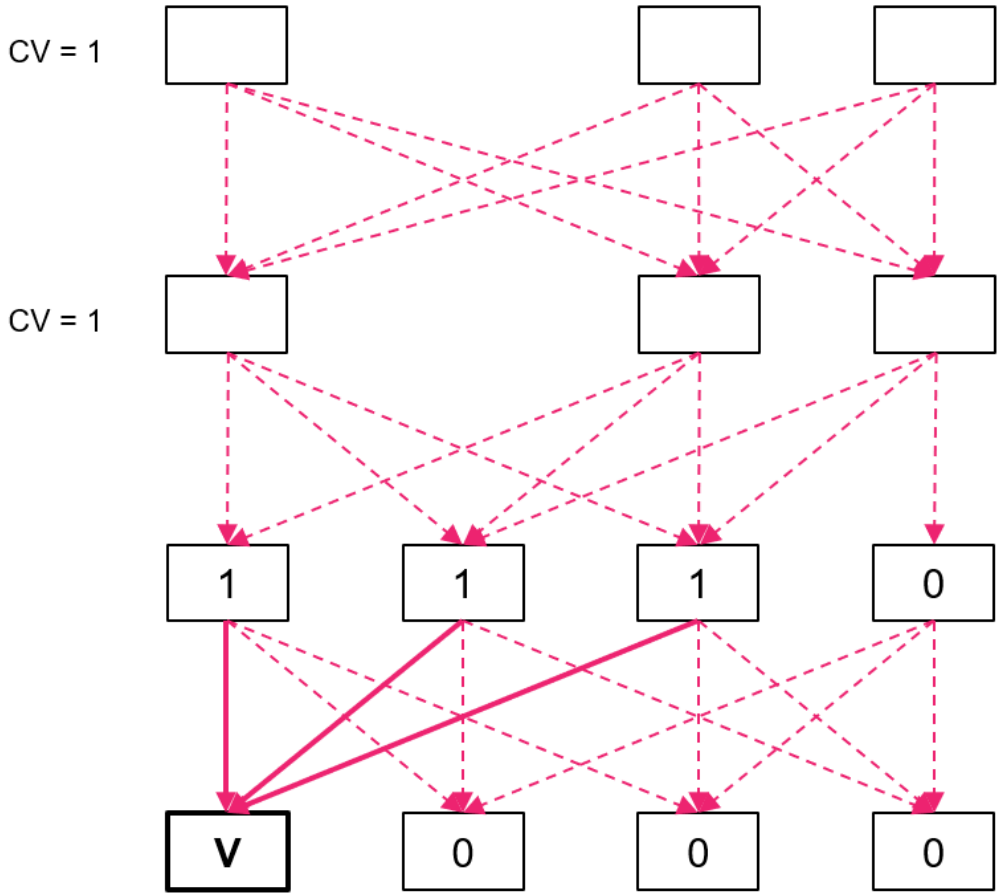
Decide

Intuition: once every unit in a round will vote for the same option, it will be decided soon

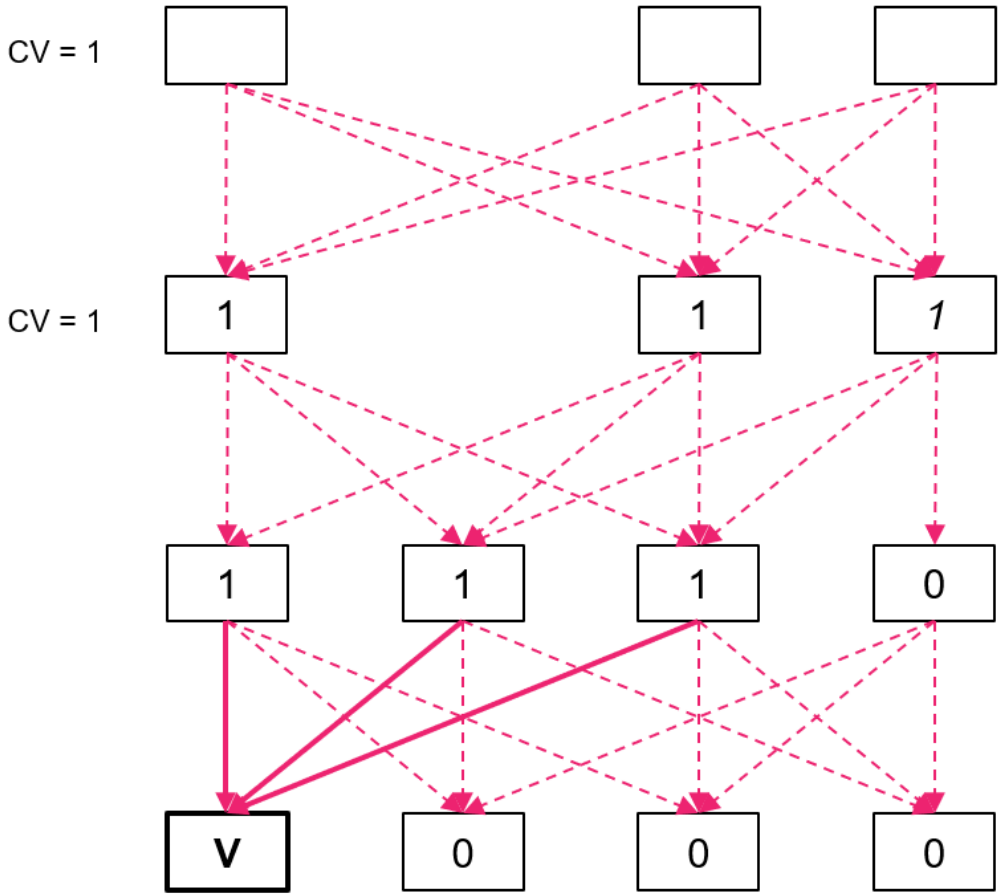
Consensus on availability - example



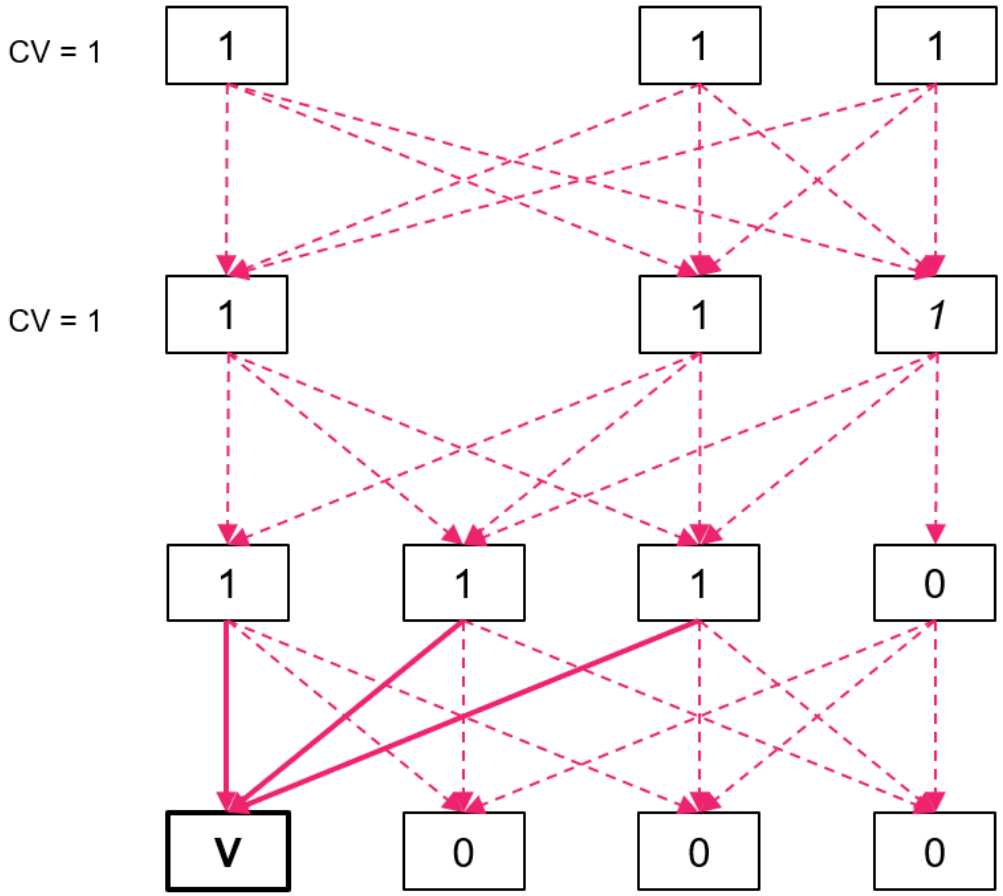
Consensus on availability - example



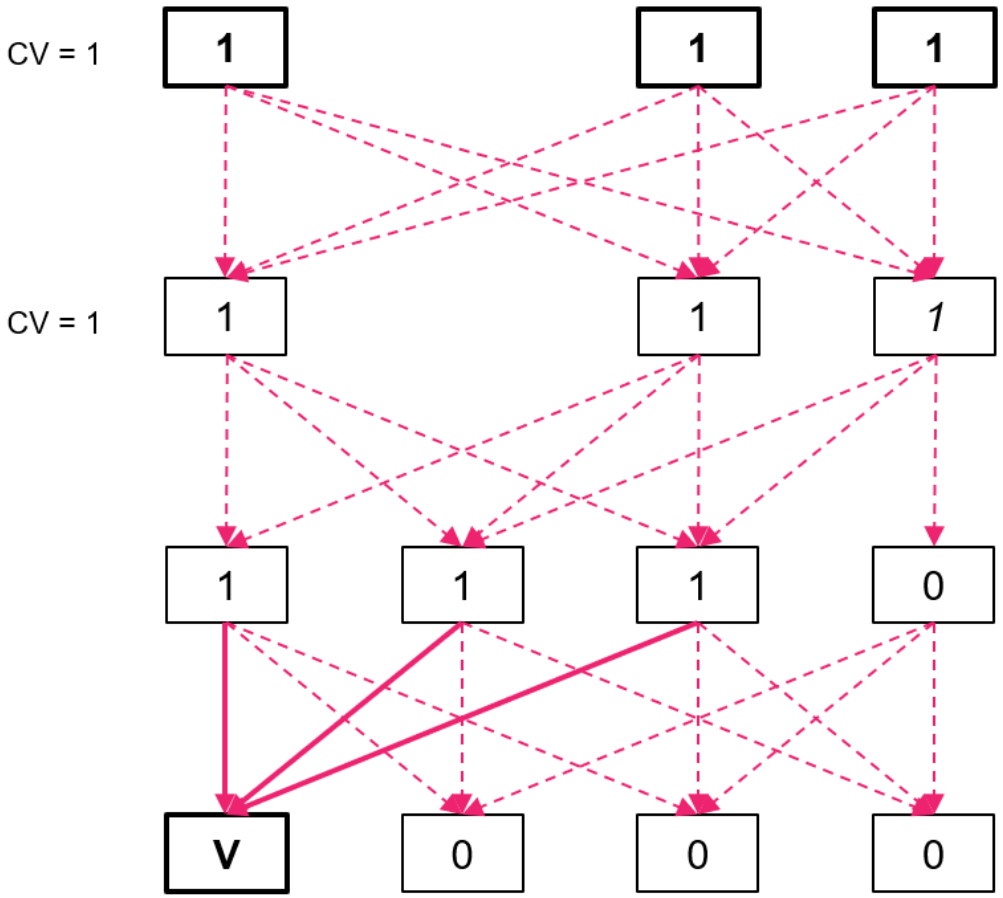
Consensus on availability - example



Consensus on availability - example



Consensus on availability - example



References

Adam Gagol, Michał Świetek (2018)

Aleph: A Leaderless, Asynchronous, Byzantine Fault Tolerant Consensus Protocol

Adam Gagol, Damian Leśniak, Damian Straszak, Michał Świetek (2019)

Aleph: Efficient Atomic Broadcast in Asynchronous Networks with Byzantine Nodes

Adam Gagol, Damian Straszak (2020)

Blockchain Fundamentals

The End