# Computer Systems Summary

## Contents

## Consensus

There are $n$ nodes, of which at most $f$ might crash. Node $i$ starts with input $v_i$. The nodes must decide for one of those values, statisfying the following properties:

1. **Agreement**: All correct nodes decide for the same value.
2. **Termination**: All correct nodes terminate in finite time.
3. **Validity**: The decision value must be the input value of a node.

### Impossibility of Consensus

There is no deterministic algorithm which always achieves consensus in the asynchronous model with $f > 0$.

## Byzantine Agreement

Finding consensus in a system with byzatine nodes is called byzantine agreement. An algorithm is $f$-resilient if it still works with $f$ byzantine nodes.

### Byzantine

A node which can have arbitrary behavior is called byzantine.

### Validity

**Any-Input Validity** The decision value must be the input value of any node.

**Correct-Input Validity** The decision value must be the input value of a correct node.

**All-Same Validity** If all correct nodes start with the same input $v$, the decision value must be $v$.

**Median Validity** If the input values are orderable, byzantine outliers can be prevented by agreeing on a value close to the median of the correct input values.

## Consistency

### Overview

| Consistency Model | Implies | Composable |
|---|---|---|
| Linearizability | Sequential Consistency, Quiescent Consistency | yes |
| Sequential Consistency | Happened-Before Consistency | no |
| Happened-Before Consistency | Sequential Consistency | no |
| Quiescent Consistency | | |

### Sequential Execution

No two operations are concurrent, we have either $f < g$ or $g < f$.

### Restricted Execution

For some object $o$ and some execution $E$, the *restricted execution* $E|o$ is $E$ filtered to only contain operations involving the $o$.

### Composability

A consistency model is composable if for every object $o$ in the restricted execution $E|o$ is consistent, then also $E$ is consistent.

### Semantic Equivalence

Executions contain exactly the same operations and each pair of operations has the same effect in both executions.

### Linearizability

An execution $E$ is *linearizable* if there exists a sequential execution $S$ such that:

1. $S$ is correct and sematically equivalent to $E$.
2. Whenever $f < g$ in $E$, then $f < g$ in $S$.

Linearizability is composable.

A system is linearizable if every possible execution is linearizable.

### Sequential Consistency

An execution $E$ is *sequentially consistent* if there exists a sequential execution $S$ such that:

1. $S$ is correct and sematically equivalent to $E$.
2. Whenever $f < g$ *on the same node* in $E$, then $f < g$ in $S$.

Every linearizable execution is sequentially consistent.

Sequential consistency is not composable.

### Quiescent Consistency

An execution $E$ is *quiescently consistent* if there exists a sequential execution $S$ such that:

1. $S$ is correct and sematically equivalent to $E$.
2. Let $t$ be some quiescent point, meaning for all operations $f$ we have $f_\dagger < t$ or $f_* > t$. Then for every $t$ and every pair of operations where $g_\dagger < t$ and $h_* > t$, we have $g < h$.

Every linearizable execution is quiescently consistent.

### Happened-Before Consistency

Same as sequential consistency.

## Quorum Systems

### Access Strategy

An *access strategy* $Z$ defines the probability $P_Z(Q)$ of accessing a quorum $Q \in S$ such that $\sum_{Q \in S} P_Z(Q) = 1$.

### Load

The *load* of access strategy $Z$ on a node $v_i$ is $L_Z(v_i) = \sum_{Q \in S_i, v_i \in Q} P_Z(Q)$.

The *load* induced by access strategy $Z$ on a quorum system $S$ is the maximal load induced by $Z$ on any node in S, which is $L_Z(S) = \max_{v_i \in S} L_Z(v_i)$.

The *load* of a quorum system $S$ is $L(S) = \min_Z L_Z(S)$.

### Work

The *work* of a quorum $Q \in S$ is the number of nodes in $Q$, $W(Q) = |Q|$.

The *work* induced by access strategy $Z$ on a quorum system $S$ is the expected number of nodes accessed, which is $W_Z(S) = \sum_{Q \in S} P_Z(Q) W(Q)$.

The *work* of a quorum system $S$ is $W(S) = \min_Z W_Z(S)$.

### Resilience

If any $f$ nodes from a quorum system $S$ can fail such that there is still a quorum $Q \in S$ without failed nodes, then $S$ is $f$-resilient.

## Game Theory

### Social Optimum

A strategy which minimizes the sum of all costs.

### Dominant Strategy

A strategy is dominant if a player is never worse of by playing this strategy.

### Nash Equilibrium

A strategy in which no player can improve by unilaterally changinh its strategy.

### Mixed Nash Equilibrium

A strategy in which at least one player is playing a randomized strategy, and no player can improve their expected payoff by unilateraly changing their strategy.

### Price of Anarchy

Let $NE_-$ denote the Nash Equilibrium with the highest cost. The price of anarchy is defined as $PoA = \frac{\text{cost}(NE_-)}{\text{cost}(SO)}$.

### Optimistic Price of Anarchy

Let $NE_+$ denote the Nash Equilibrium with the smallest cost. The optimistic price of anarchy is defined as $OPoA = \frac{\text{cost}(NE_+)}{\text{cost}(SO)}$.

## File System

The filing system virtualizes the collection of storage devices in the system:

- **Multiplexing**: Sharing the storage between applications and users.
- **Abstraction**: Making the devices appear as a more convenient collectionof files with consistency properties.
- **Emulation**: Creating this illusion over an arbitrary set of storage devices.