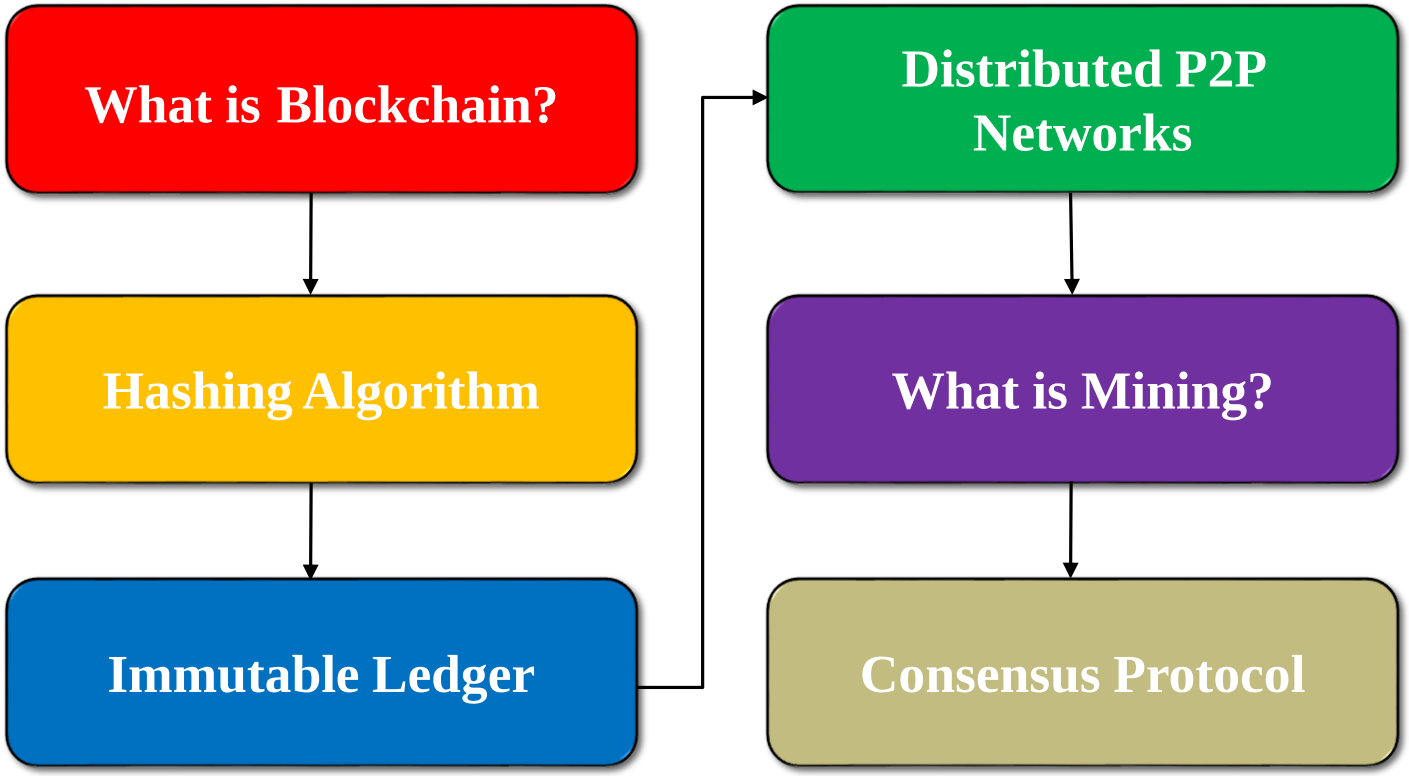
Blockchain

Dr. Bahar Ali
Assistant Professor (CS), National University Of Computer and Emerging Sciences, Peshawar.

Contents – Module A

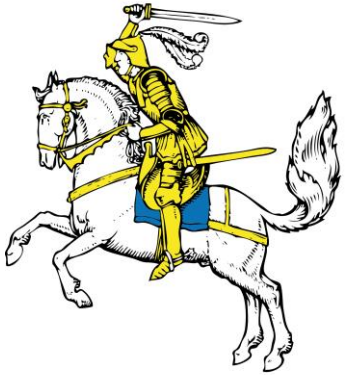


Byzantine Generals Problem

Byzantine Generals Problem

- The Byzantine Generals' Problem', **published in 1982**
- Describes **the difficulty a decentralized systems have in agreeing on a single truth**
- The term takes its name from a story, the "Byzantine generals problem", developed to describe a situation in which, in order to avoid catastrophic failure of the system, the system's actors must agree on a concerted strategy, but some of these actors are unreliable.

Byzantine Generals Problem



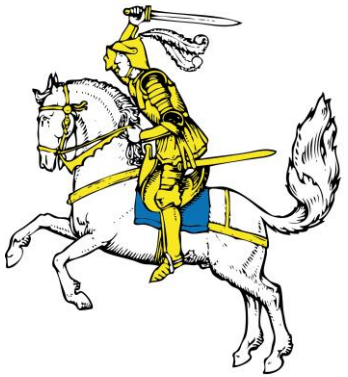
Byzantine Generals Problem

- Generals from different locations want to attack a palace.
- No king, so they have to decide, whether to attack or not
- If all are loyal and trust each other, they will agree at time of attack and can easily win. However,
- The problem is when some of them are traitors, then what should be done?

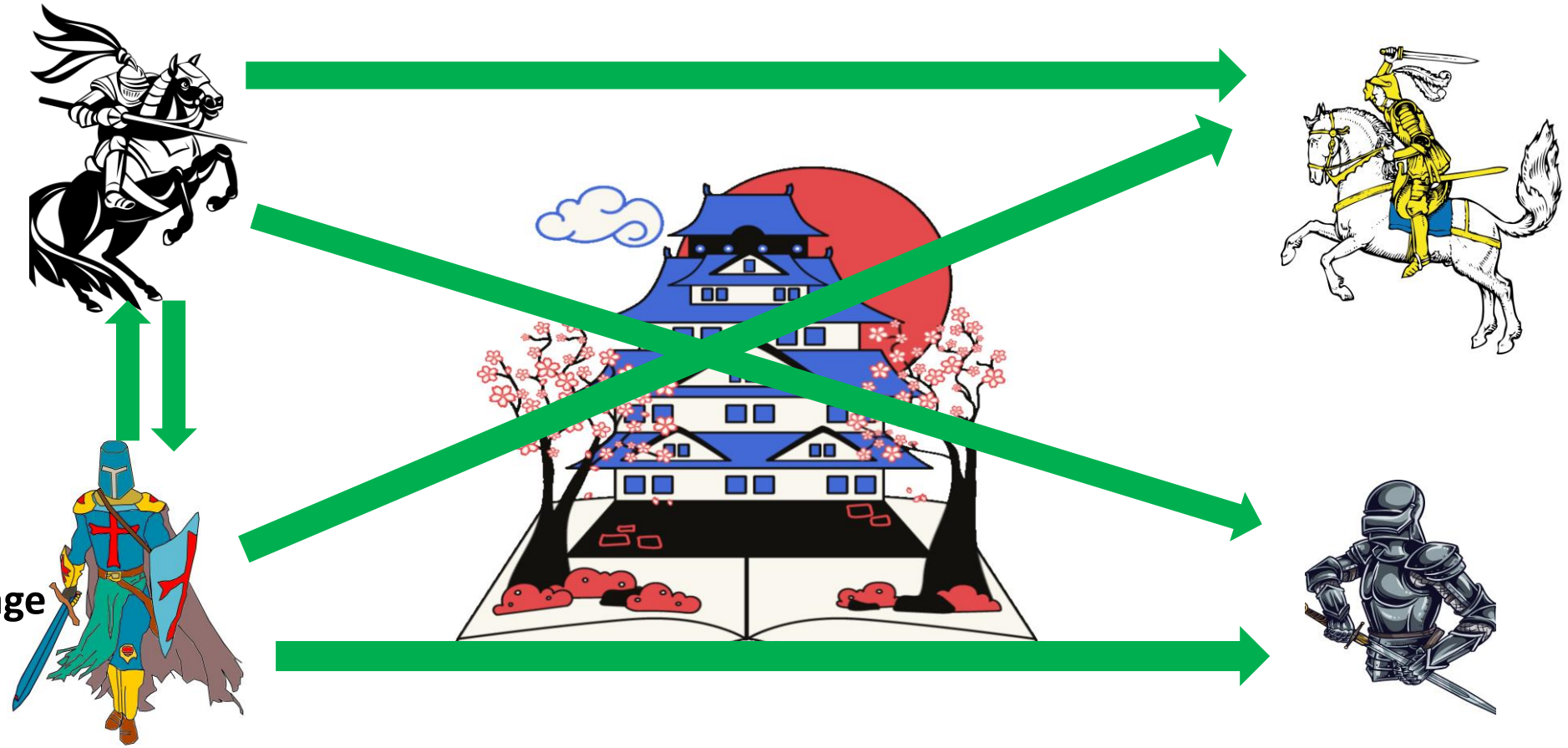
- Same in the case of a distributed network, when hackers are involved, then what should be done?
- How to smoothly run the consensus protocol, in case of Blockchain?

Byzantine Generals Problem

He sends message to attack

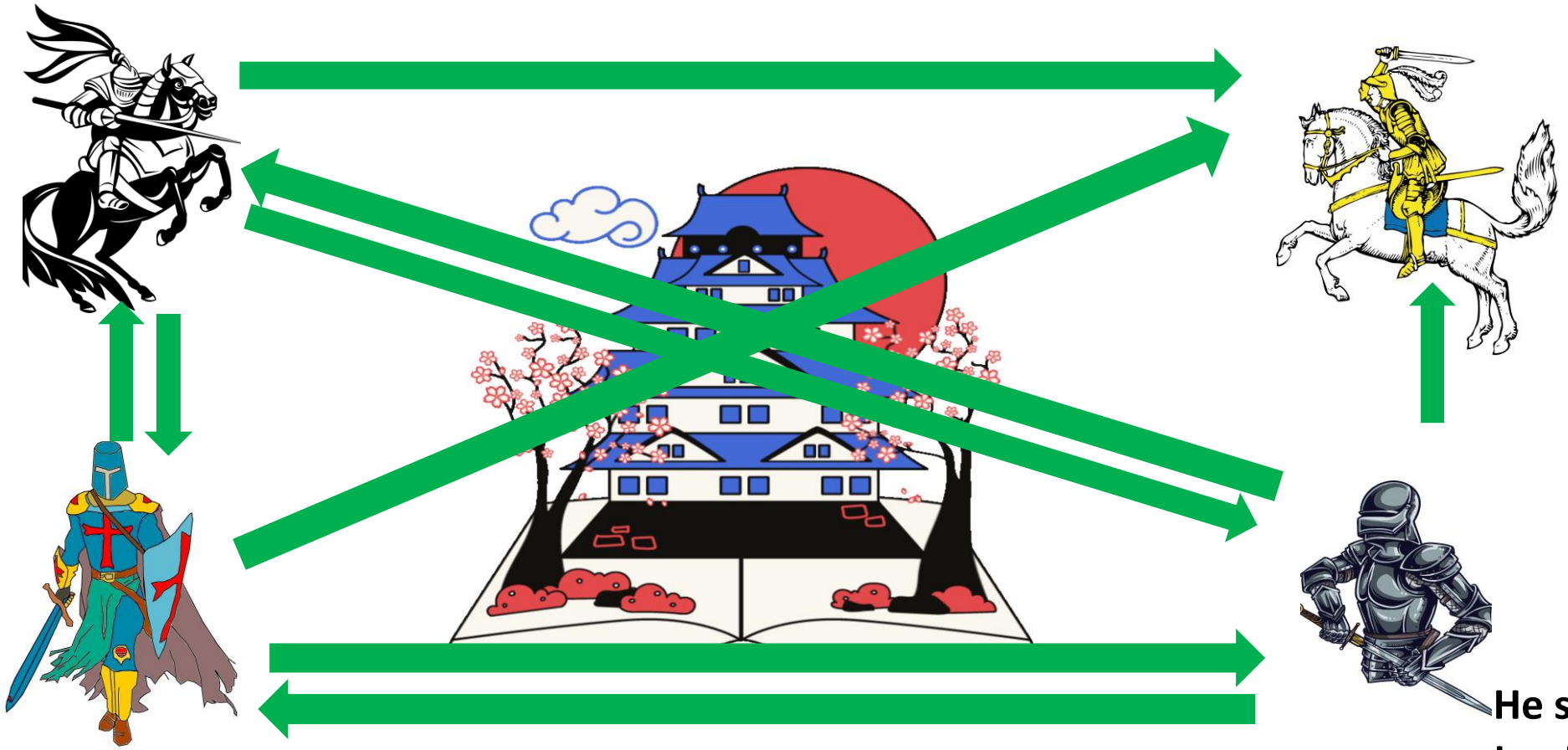


Byzantine Generals Problem



He sends message to attack

Byzantine Generals Problem



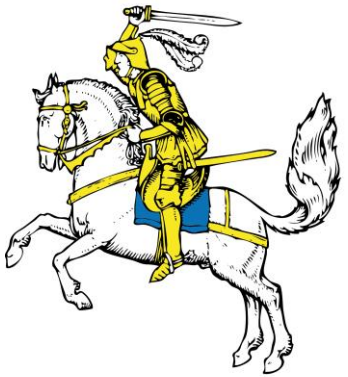
He sends message to attack

Byzantine Generals Problem

He sends message to attack

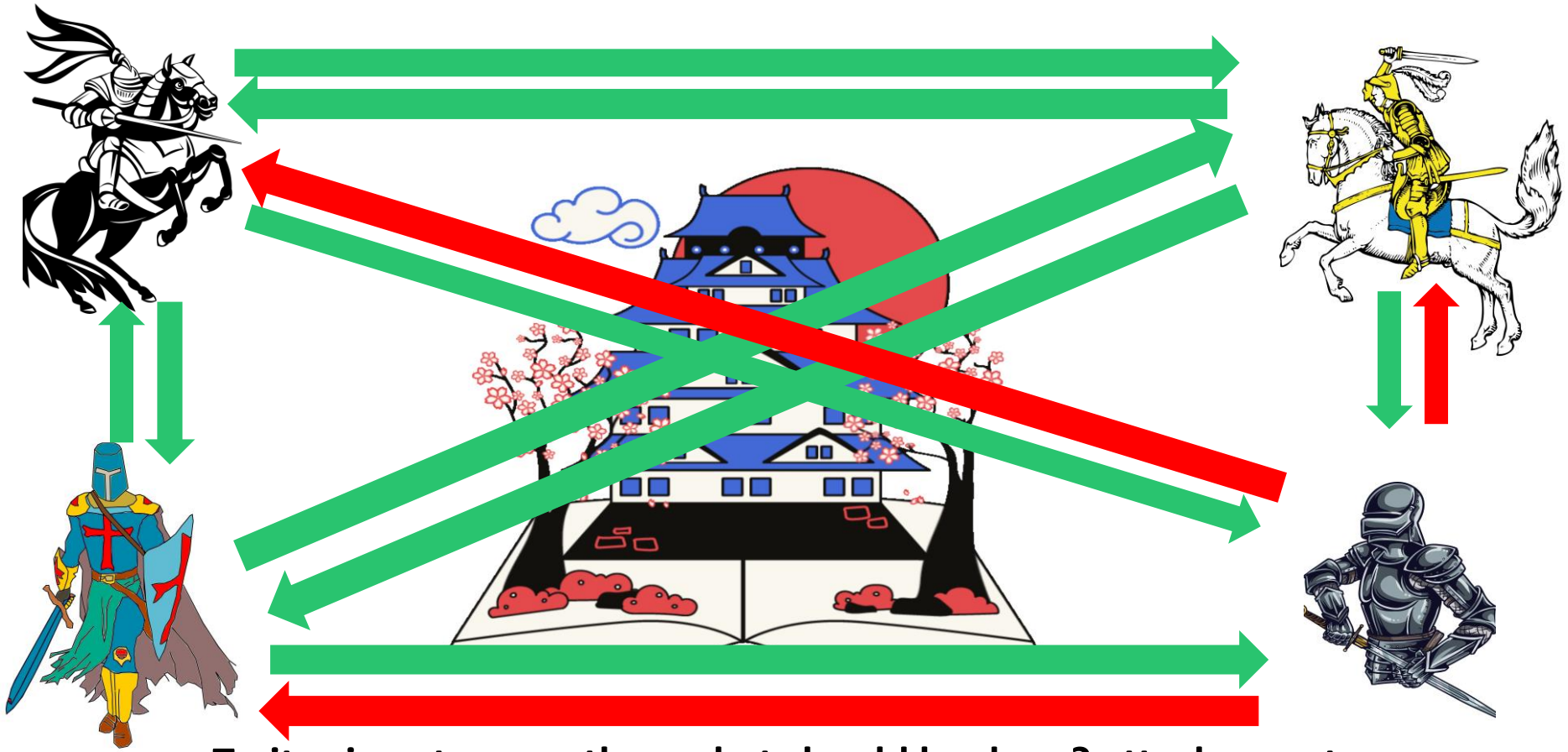
All agree at a time and easily win

Byzantine Generals Problem



Traitor

Byzantine Generals Problem



Traitor

Traitor is not agree, then what should be done? attack or not

Practical Byzantine Fault Tolerance

by
Miguel Castro

Abstract

Our growing reliance on online services accessible on the Internet demands highly-available systems that provide correct service without interruptions. Byzantine faults such as software bugs, operator mistakes, and malicious attacks are the major cause of service interruptions. This thesis describes a new replication algorithm, BFT, that can be used to build highly-available systems that tolerate Byzantine faults. It shows, for the first time, how to build Byzantine-fault-tolerant systems that can be used in practice to implement real services because they do not rely on unrealistic assumptions and they perform well. BFT works in asynchronous environments like the Internet, it incorporates mechanisms to defend against Byzantine-faulty clients, and it recovers replicas proactively. The recovery mechanism allows the algorithm to tolerate any number of faults over the lifetime of the system provided fewer than 1/3 of the replicas become faulty within a small window of vulnerability.

Byzantine Generals Problem

Q) How this Byzantine Fault Tolerance works in Blockchain?

A) we will look into it in the consensus protocol

Consensus Protocol

Consensus Protocol

- Covered the following topics
- Base for consensus protocol

  - **Hashing Algorithm** 
  - **Immutable Ledger** 
  - **Mining** 
  - **Distribute P2P network** 

Consensus Protocol

Consensus Protocol helps:

1. To **Prevent Attacks**
2. To Solve **Competing Chain Problem**

Consensus Protocol

Prevent Attacks

Consensus Protocol (Prevent Attacks)

- **To Prevent attacks on Blockchain**

Types of Consensus Protocol:

- **Proof of Work (POW) (We will discuss this one)**
- **Proof of Stake (POS)**
- **Others**

Consensus Protocol

- Anyone (miner) can add a new block in a Blockchain
- A hacker can insert a malicious node at the end of a chain
- The block is new, so you can not say whether the block has been changed or not.

- To solve this problem a consensus protocol is used, which uses a **Proof of work (PoW)** algorithm.

- https://www.crypto51.app/

Consensus Protocol

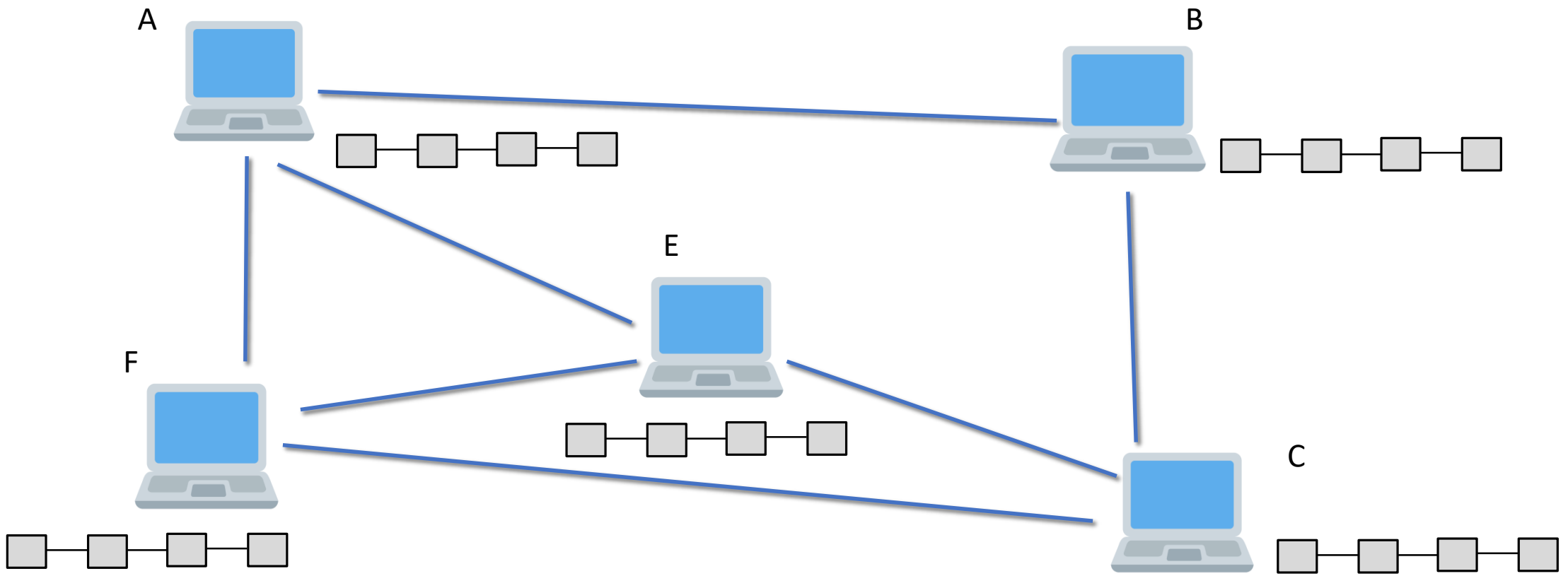
Proof of work (PoW)

- Miners give proof of work, in sense that a mathematical problem has been solved
- Takes time, consumes a lot of electricity and utilize the hardware
- Miner gets a reward for adding a new block

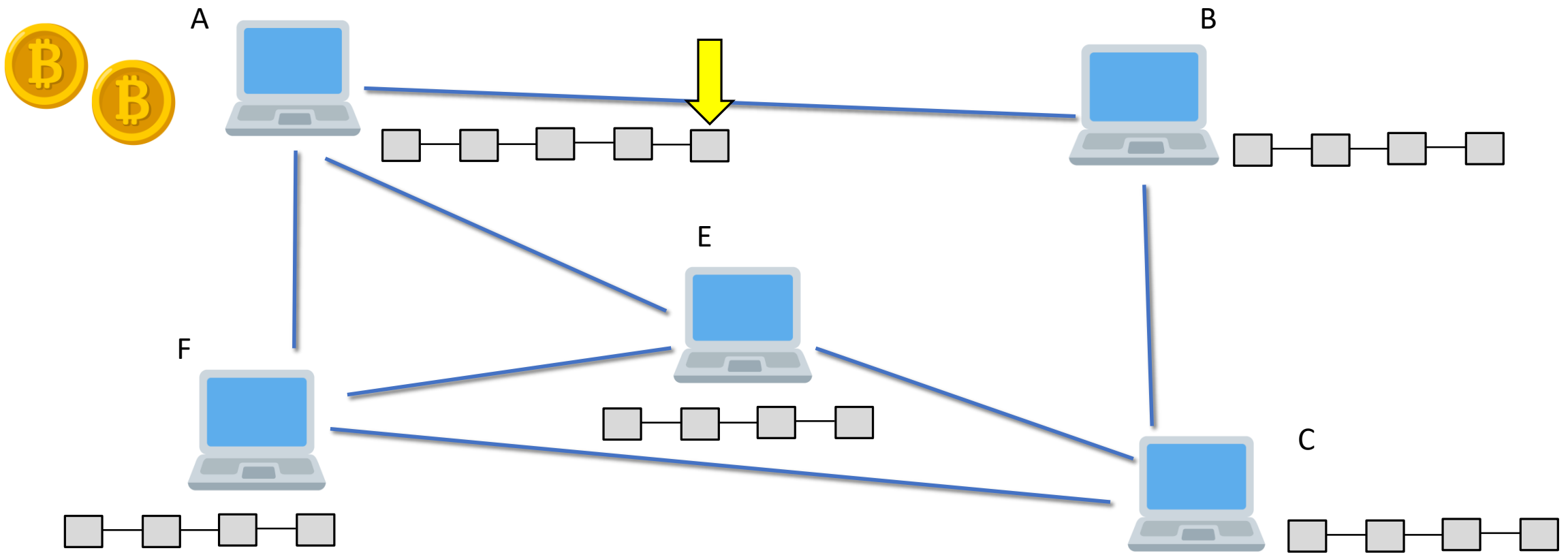
Consensus Protocol

- Miners solve a mathematical problem
- Takes time, consumes a lot of electricity and utilize the hardware
- In case of any violation, the miner will not be paid
- When a block is mined by a miner, a message is sent across the network
- Other nodes do not directly add it, rather they run an algorithm to verify the block whether the block is malicious or not.
- If malicious/ invalid, then no reward will be given
- This way a malicious activity is stopped, let’s see pictorially

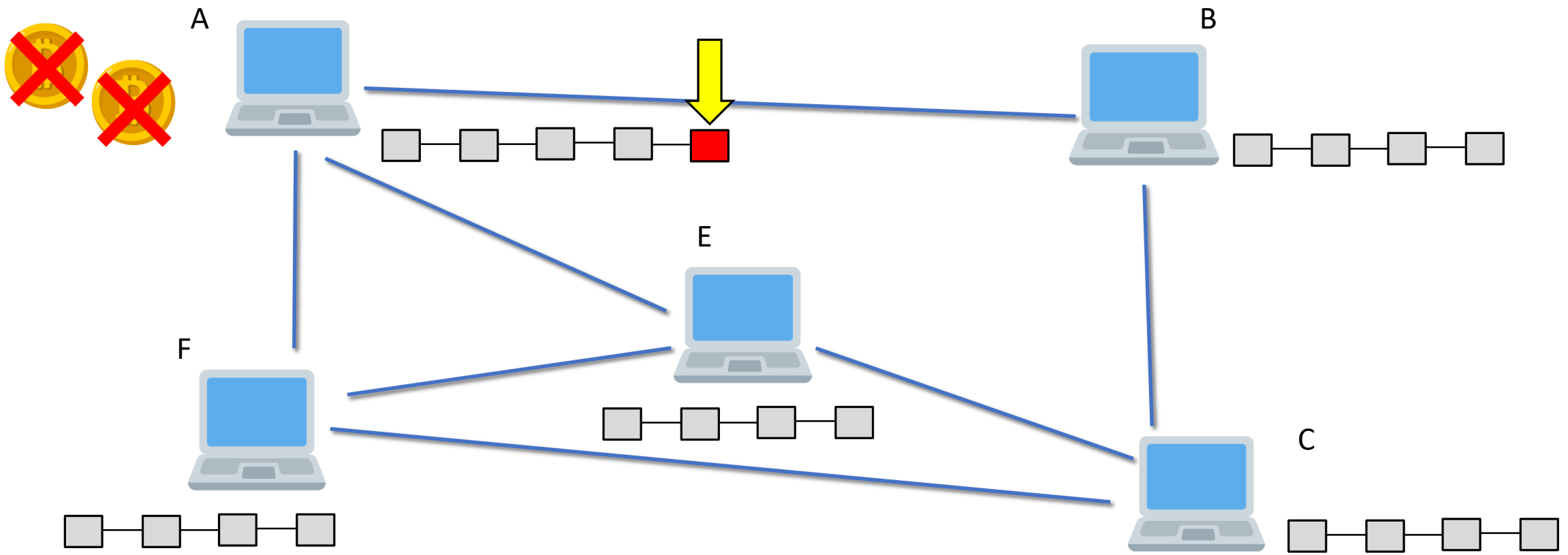
Consensus Protocol



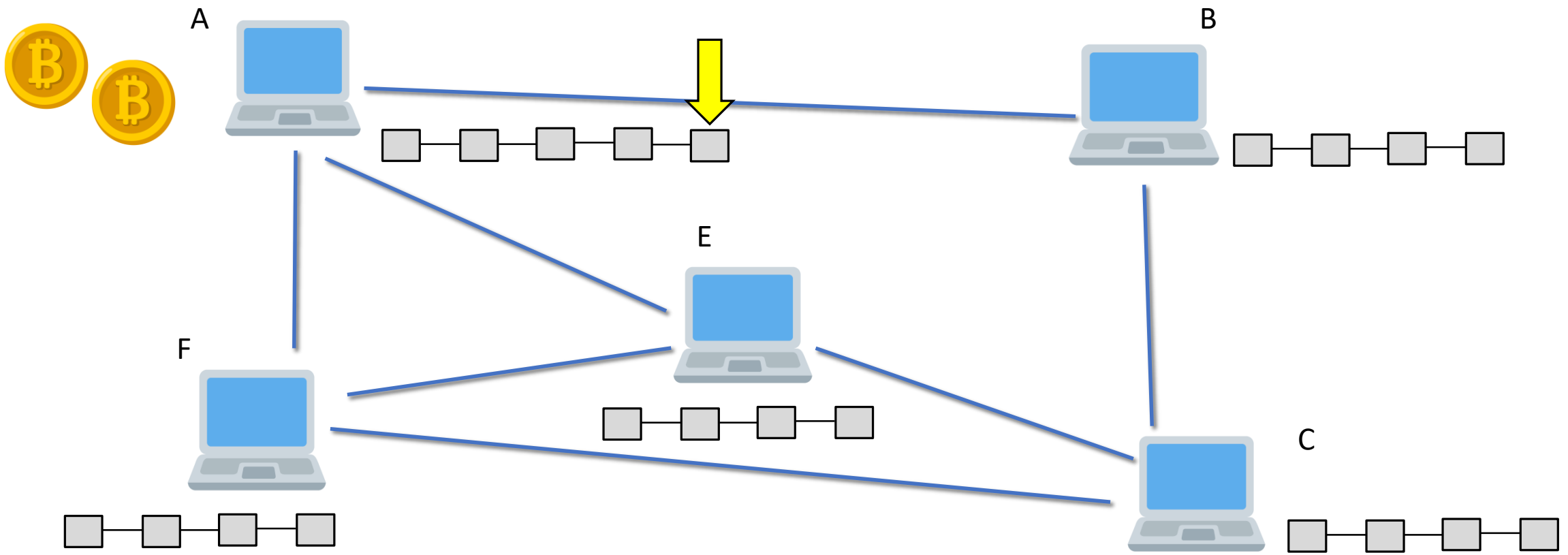
Consensus Protocol



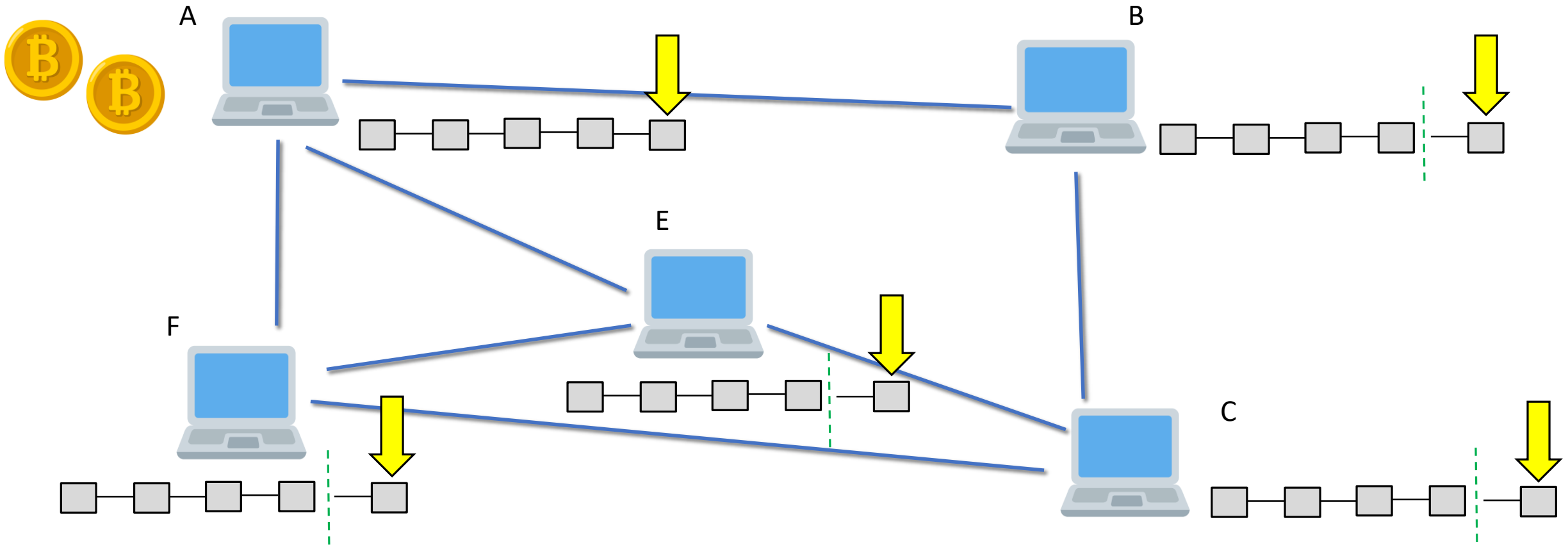
Consensus Protocol



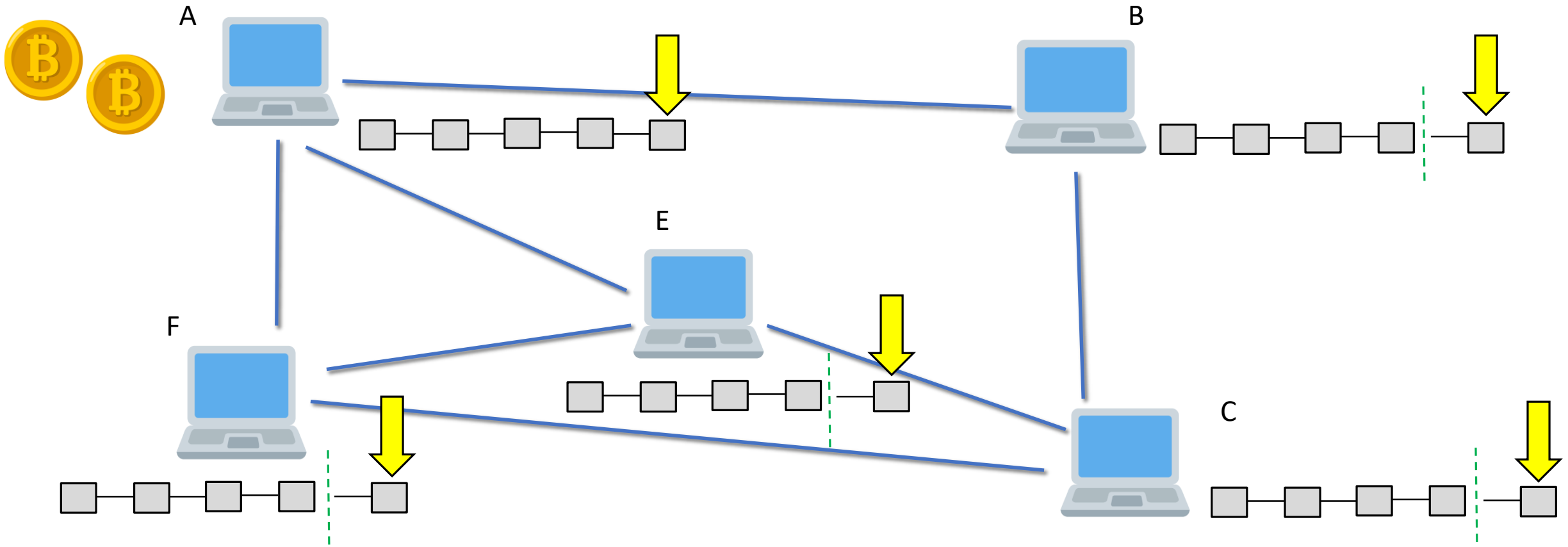
Consensus Protocol



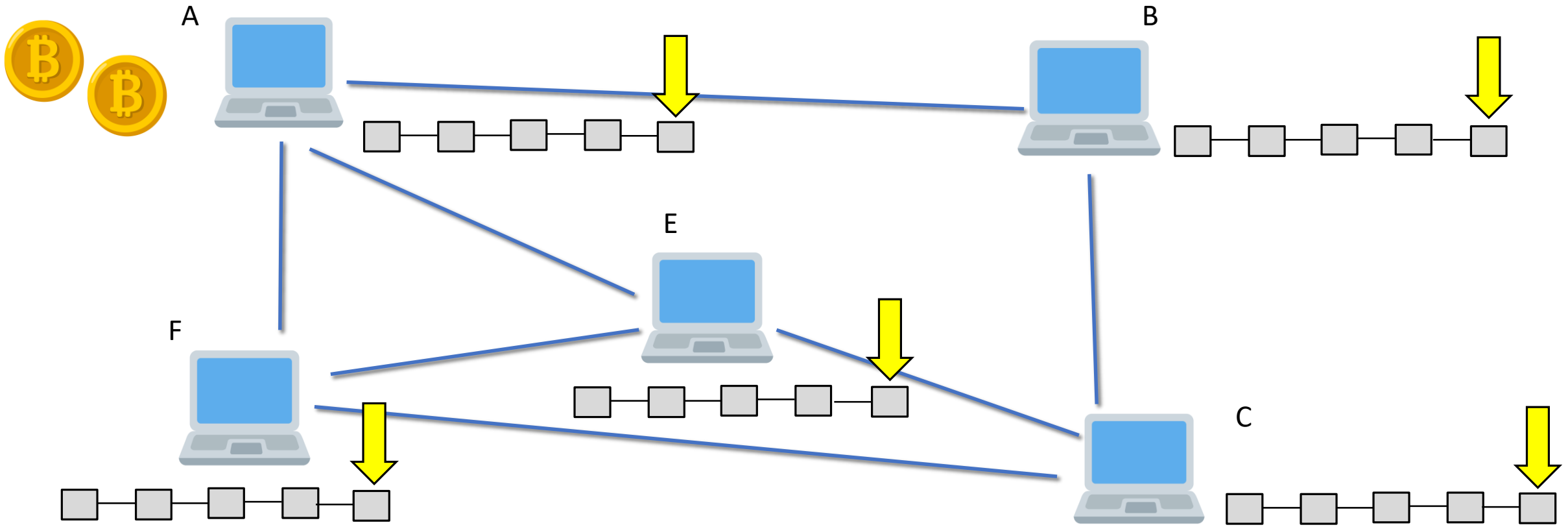
Consensus Protocol



Consensus Protocol



Consensus Protocol



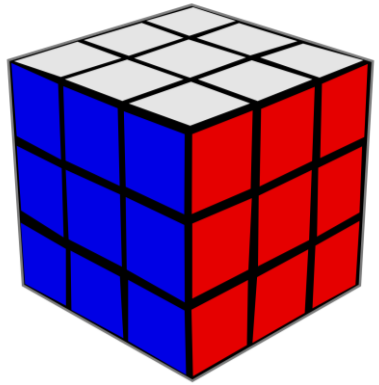
1. Check syntactic correctness
2. Reject if duplicate of block we have in any of the three categories
3. Transaction list must be non-empty
4. Block hash must satisfy claimed nBits proof of work
5. Block timestamp must not be more than two hours in the future
6. First transaction must be coinbase (i.e. only 1 input, with hash=0, n=-1), the rest must not be
7. For each transaction, apply "tx" checks 2-4
8. For the coinbase (first) transaction, scriptSig length must be 2-100
9. Reject if sum of transaction sig opcounts > MAX_BLOCK_SIGOPS
10. Verify Merkle hash
11. Check if prev block (matching prev hash) is in main branch or side branches. If not, add this to orphan block in prev chain; done with block
12. Check that nBits value matches the difficulty rules
13. Reject if timestamp is the median time of the last 11 blocks or before
14. For certain old blocks (i.e. on initial block download) check that hash matches known values
15. Add block into the tree. There are three cases: 1. block further extends the main branch; 2. blo make it become the new main branch; 3. block extends a side branch and makes it the new m
16. For case 1, adding to main branch:
1. For all but the coinbase transaction, apply the following:

Consensus Protocol

Q) Is this a time taking process ?

A) No

- Mining takes time, however, verification takes less time
- For Example, Rubik's cube solving takes much time, however, verification is easy and takes less time



Consensus Protocol

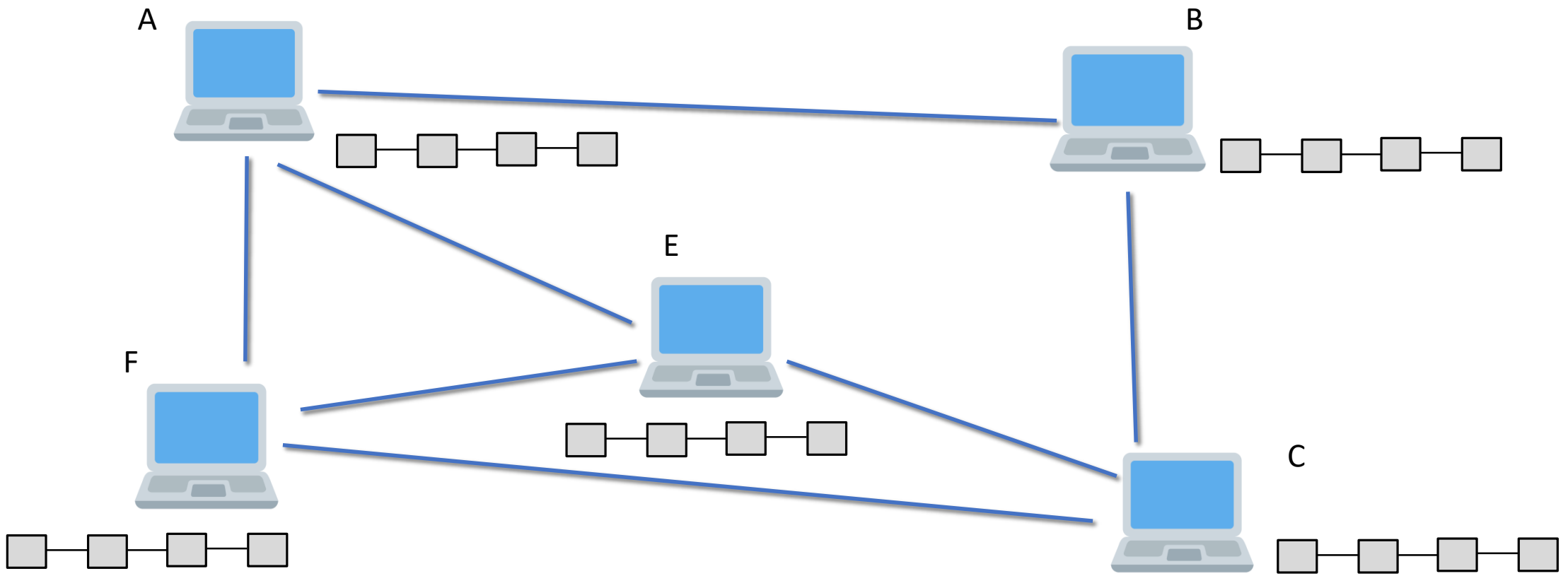
Competing Chain Problem

Consensus Protocol

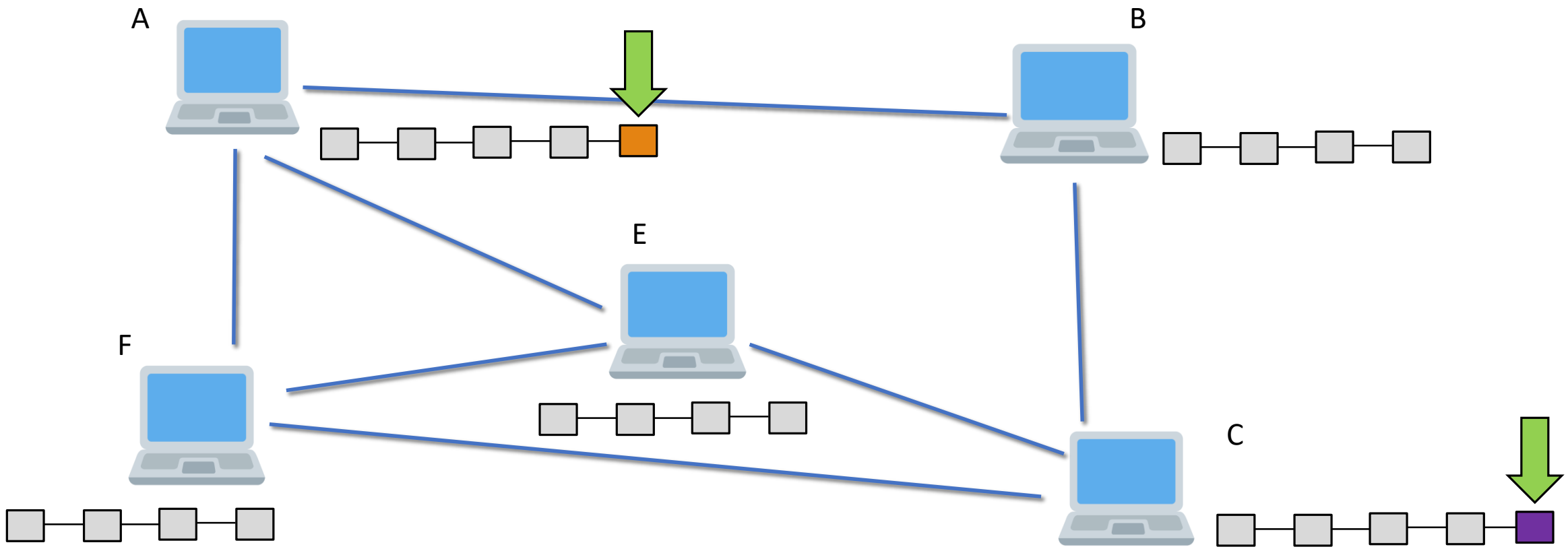
- When miners (**A** and **C**) mine block at the same time
- **A** sends information to its nearest nodes that a new block is mined
- **C** sends information to its nearest nodes that a new block is mined
- The nearest nodes verify these blocks, however, a conflict arises
- **A** and its nearest nodes have one chain, while, B and its nearest nodes have a different chain

- **How consensus protocol solves this problem?**

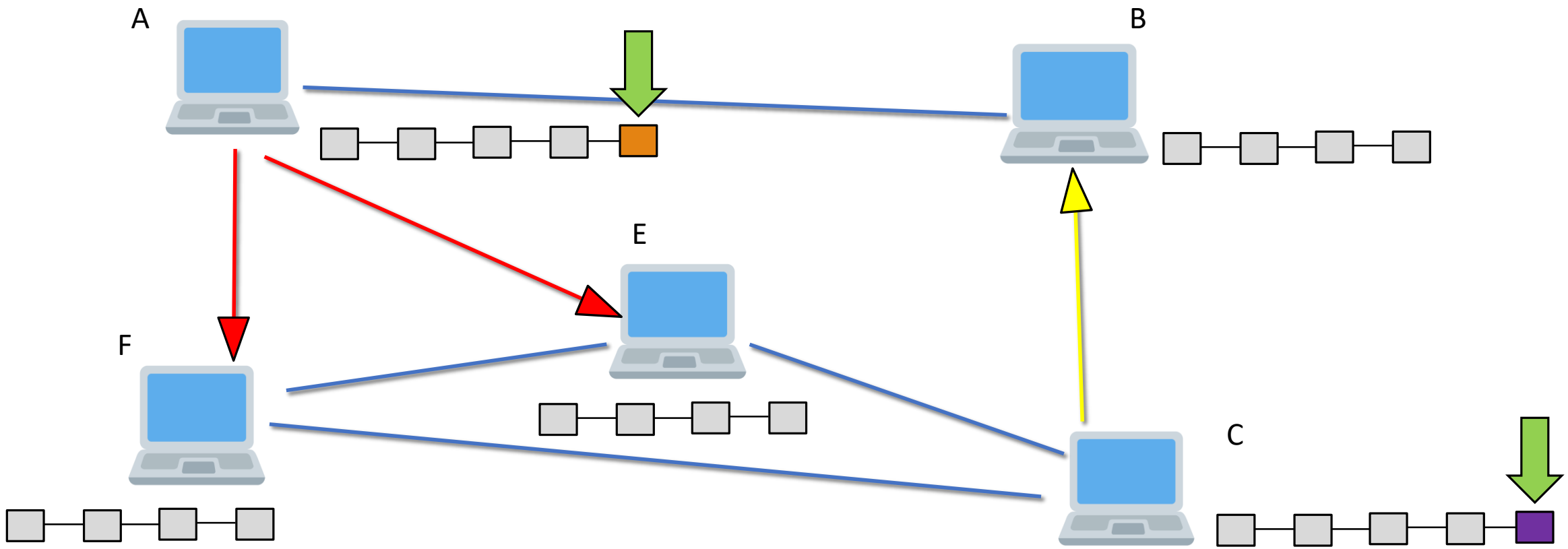
Consensus Protocol



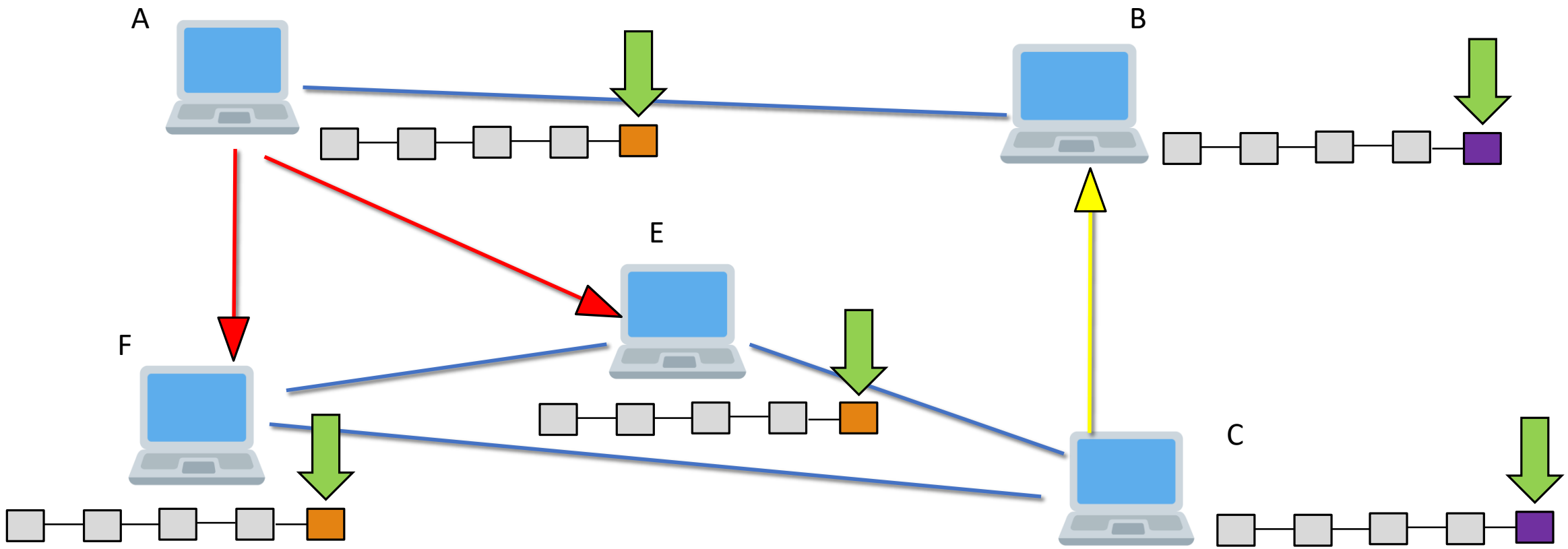
Consensus Protocol



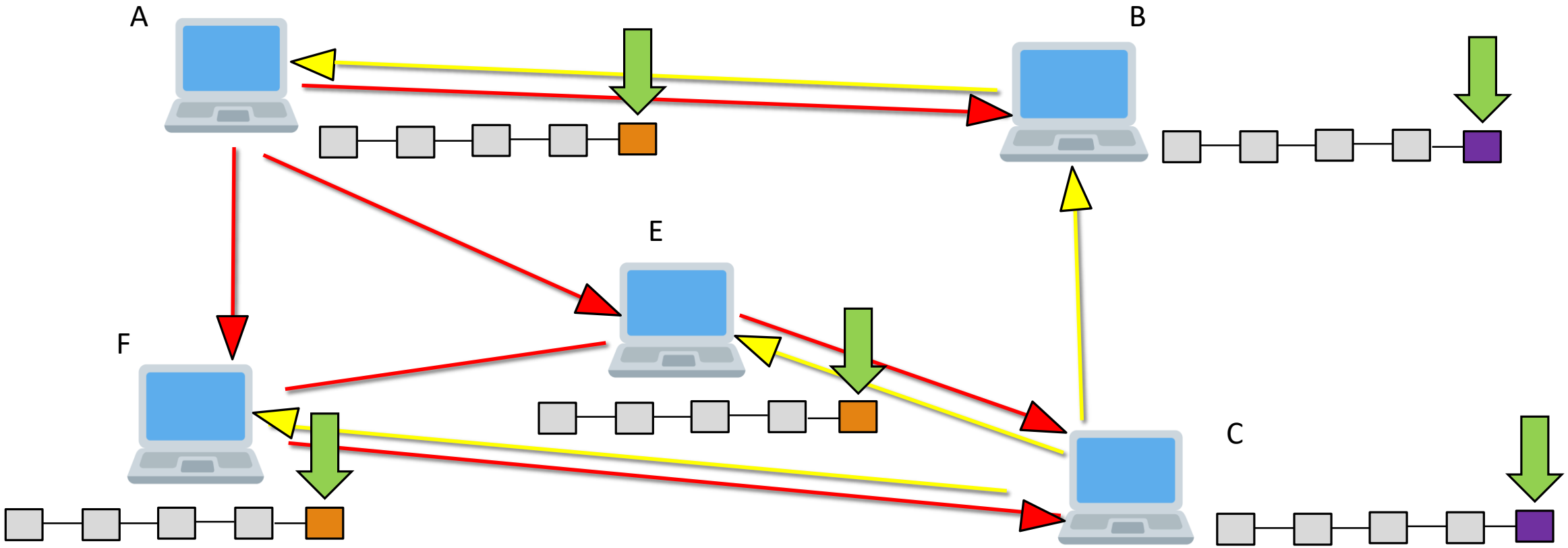
Consensus Protocol



Consensus Protocol



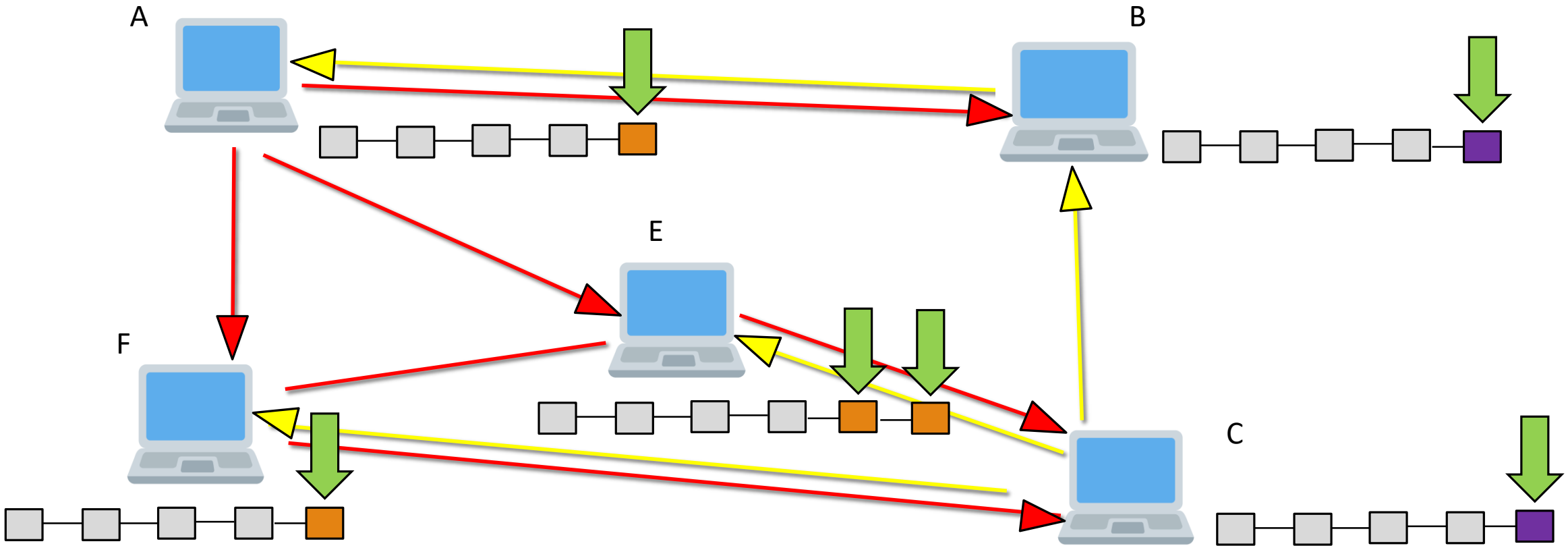
Consensus Protocol



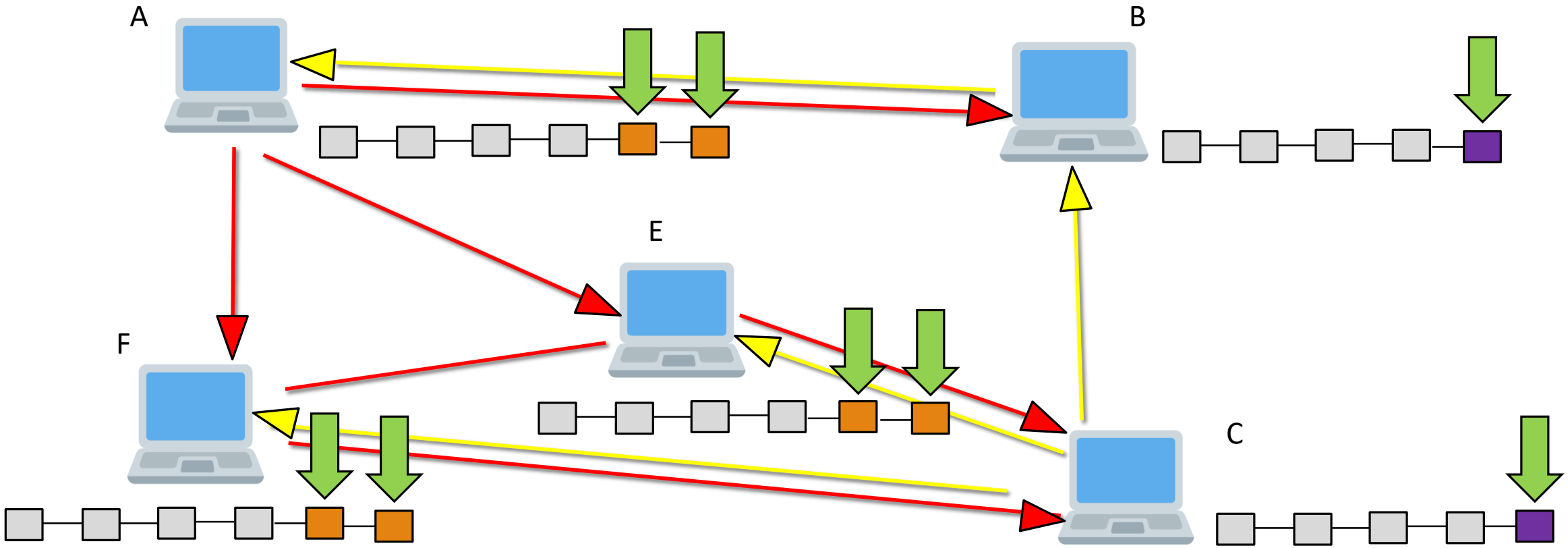
Consensus Protocol

- **How consensus protocol solves this problem?**
- Consensus protocol accepts the longer chain
- So all nodes wait until the next block is mined by the sub networks
- Generally, the sub network with high hashing power, has high probability to solve the next block quickly
- The chain of sub network which solve the next block quickly, becomes longer and is added by all the nodes
- The block which is rejected is called an orphan block

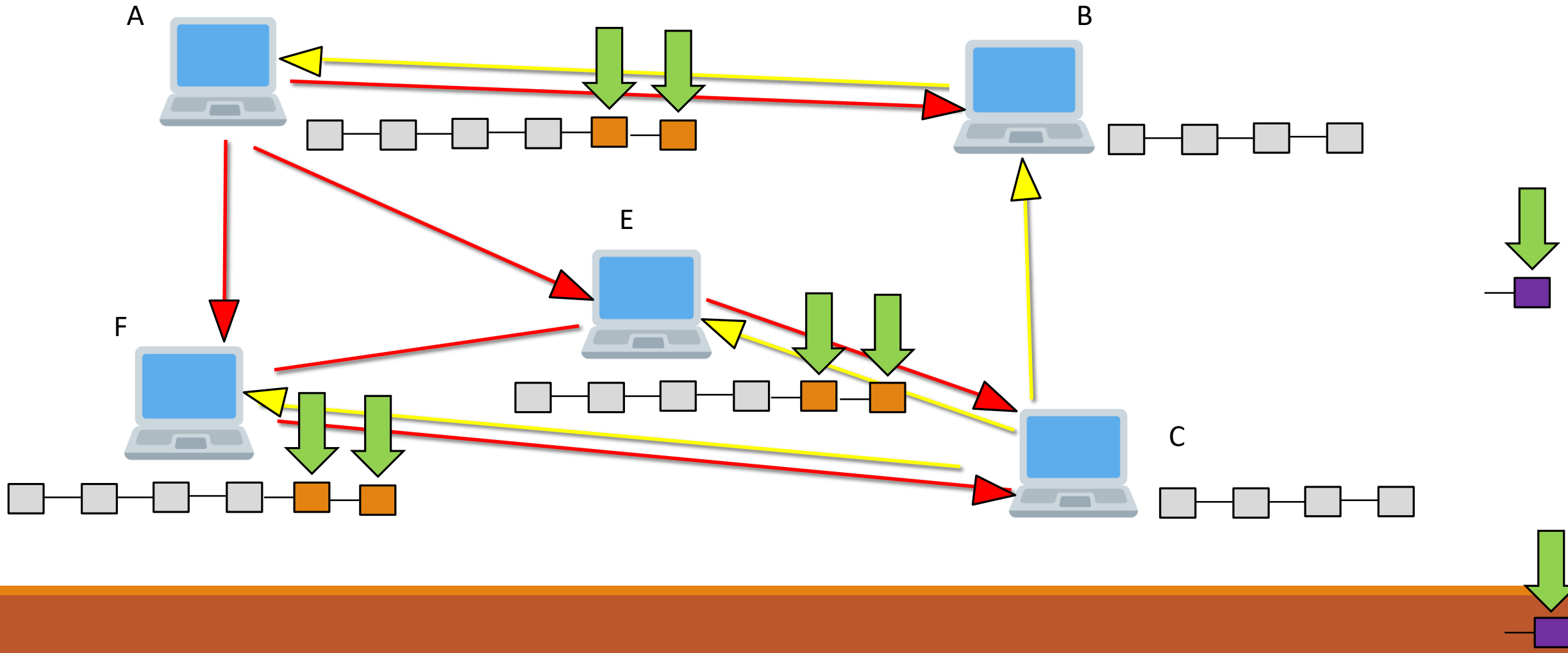
Consensus Protocol



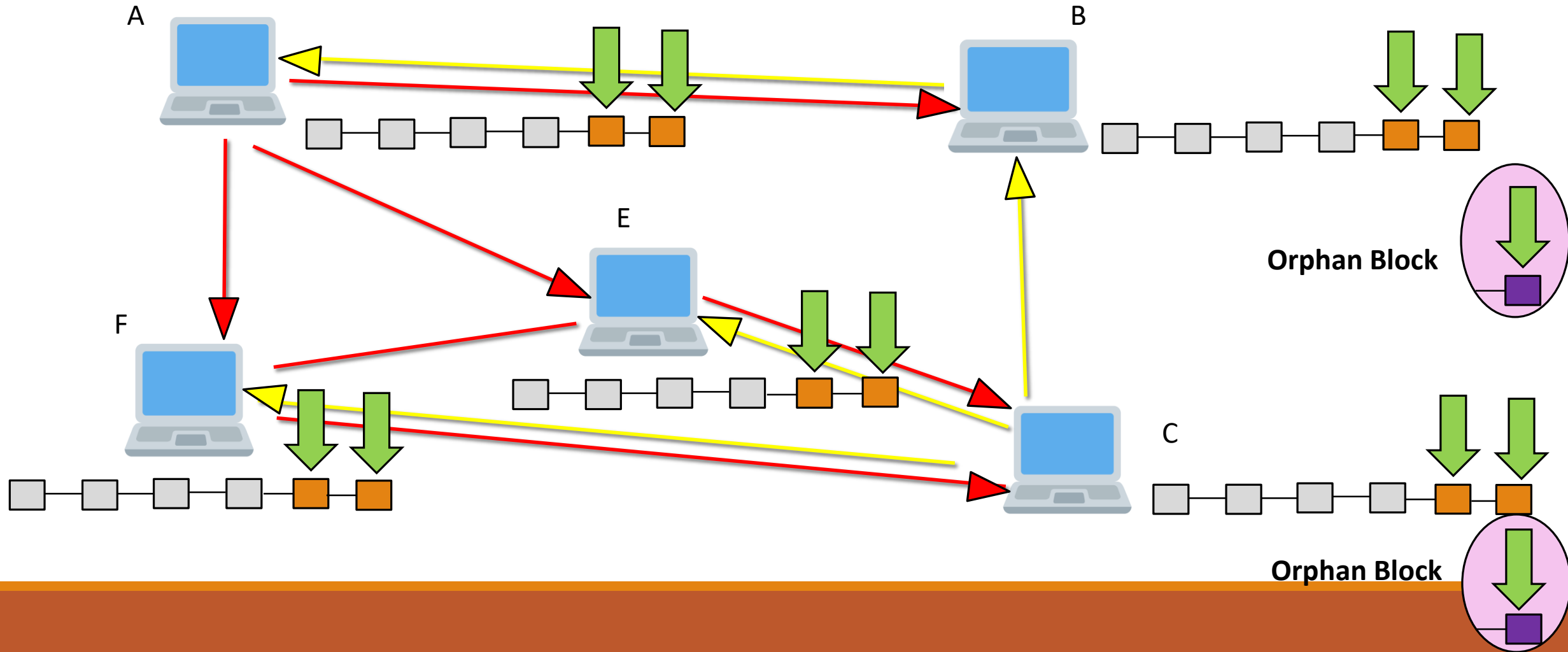
Consensus Protocol



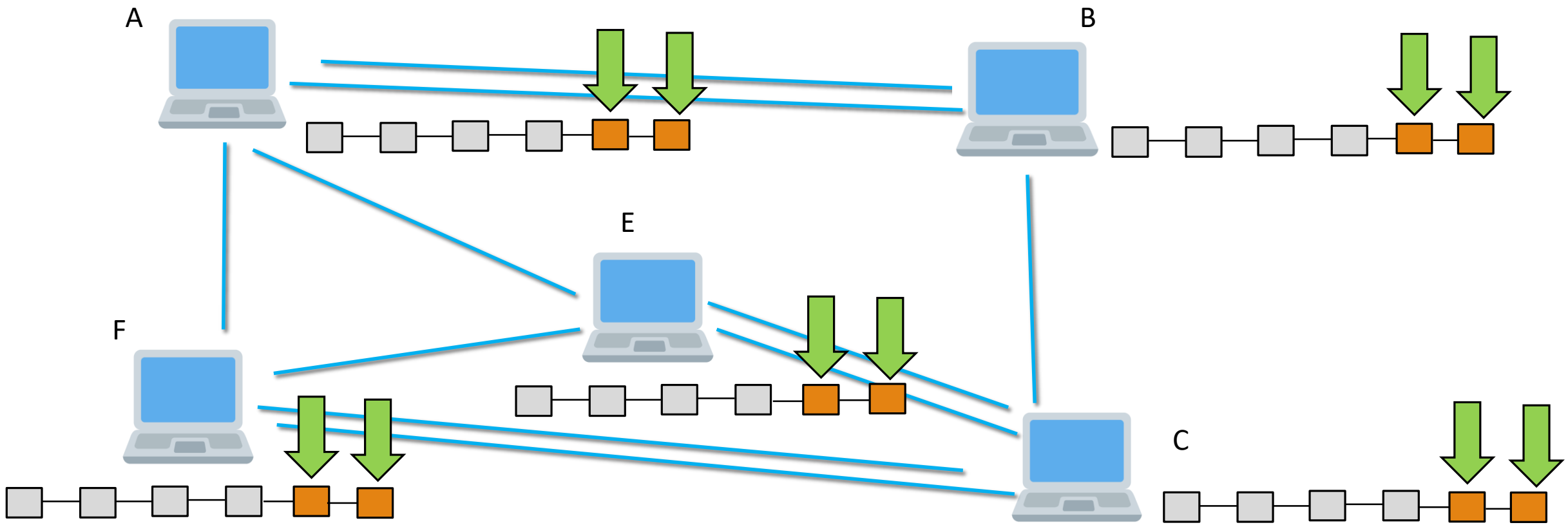
Consensus Protocol



Consensus Protocol



Consensus Protocol



Consensus Protocol

- The Consensus Protocol of Blockchain needs **51%** majority, while Byzantine Fault Tolerance needs **approximately 66%**.
- All the transaction in the **Orphan Block** will be dropped and the miner that had mined the block will not get any reward
- Therefore, wait for the **6 confirmations** before assuming payment is successful
- So it is necessary to wait until the next **five blocks** are added
- It also handle the double spending problem (we will discuss it later)
- Let’s see, how it works