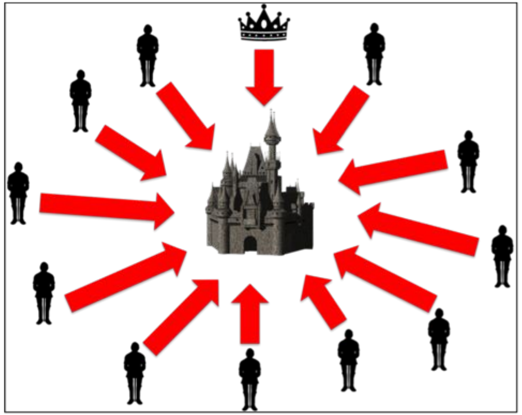
# Consensus under a Dynamic Synchronous Model

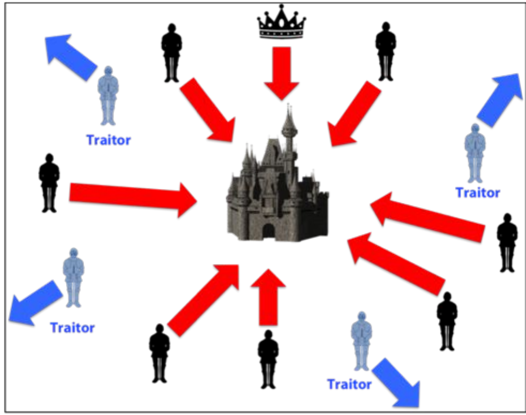
Author: Kunal Kapoor
Mentor: Jun Wan

# Byzantine Broadcast

- Background: Byzantine Generals need to attack or retreat
- Generals are split apart and communicate via messengers
- Some messengers are traitors/secret enemies
- How do they proceed?



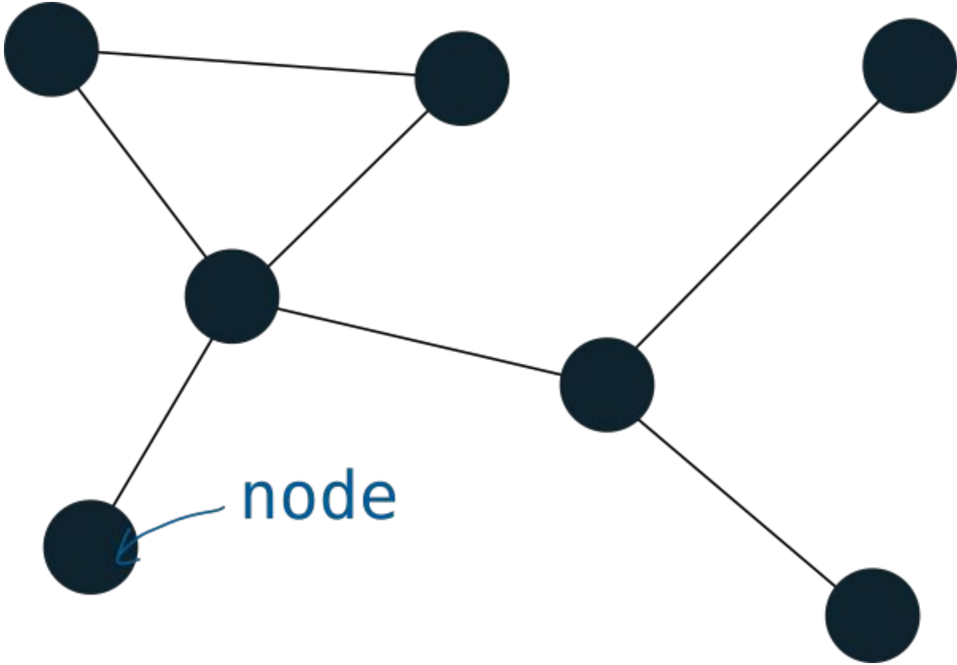
Coordinated Attack Leading to Victory



Uncoordinated Attack Leading to Defeat

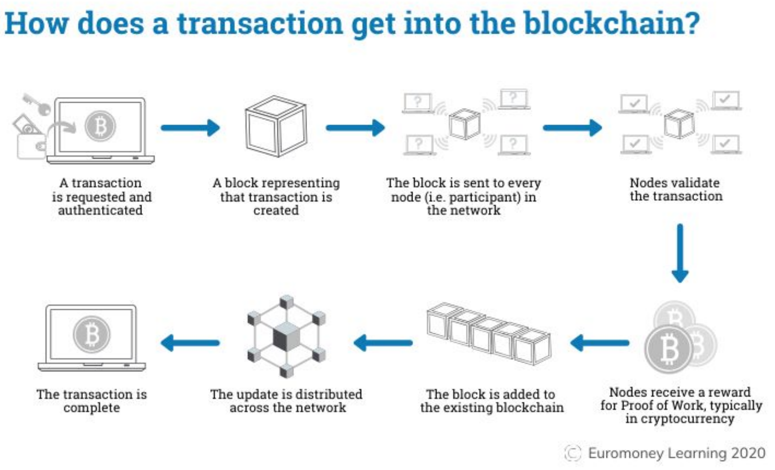
# General Problem Definition

- n users in a system
  - Honest Users
  - Corrupt Users
- GOAL: Achieve Consensus
  - Consistency
  - Validity



# Real World Applications

- Area: Distributed Computing
- Blockchain
- Other Distributed Systems



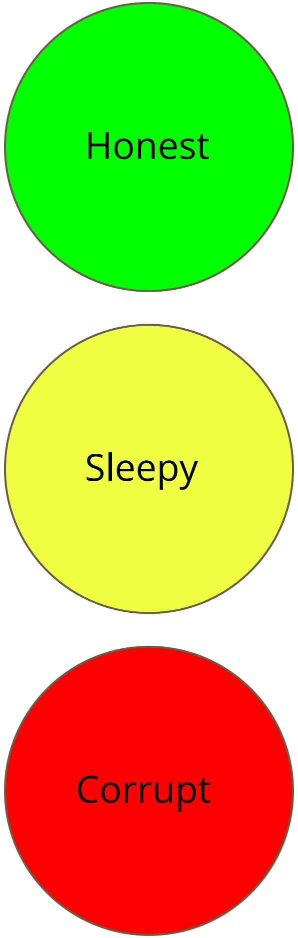
How does a transaction get into the blockchain?

A transaction is requested and authenticated
A block representing that transaction is created
The block is sent to every node (i.e. participant) in the network
Nodes validate the transaction
Nodes receive a reward for Proof of Work, typically in cryptocurrency
The block is added to the existing blockchain
The update is distributed across the network
The transaction is complete

© Euromoney Learning 2020

# Past Work

- Known n and h
  - n represents **Total Population**
  - h represents **Online Honest Population**
  - f represents **Online Corrupt Population**
- Honest Majority
  - $h > n/2$
- No Sleepy Users
  - Sleepy users can **go offline**
- Blockchain Approach vs Trust Graph



Honest

Sleepy

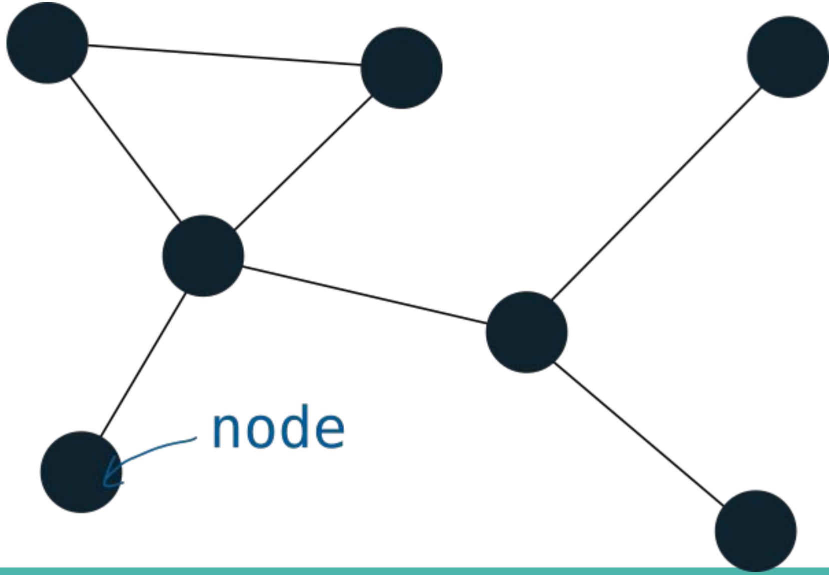
Corrupt

# Dynamic Synchronous Model

- n, h - unknown
- A constant c is known s.t. $c < h/n$
- Sleepy users
- Solution
    - 2 Building Blocks
    - Byzantine Broadcast Proper
- Main Result: Adapting the Post Processing algorithm

# Building Block 1: Trust Graph

- Graph mapping relations between users
  - Edge signifies mutual trust
- Each user has unique trust graphs
- Honest users remain connected
- Edge Removal
  - Distrust Messages
  - Equivocation Evidence

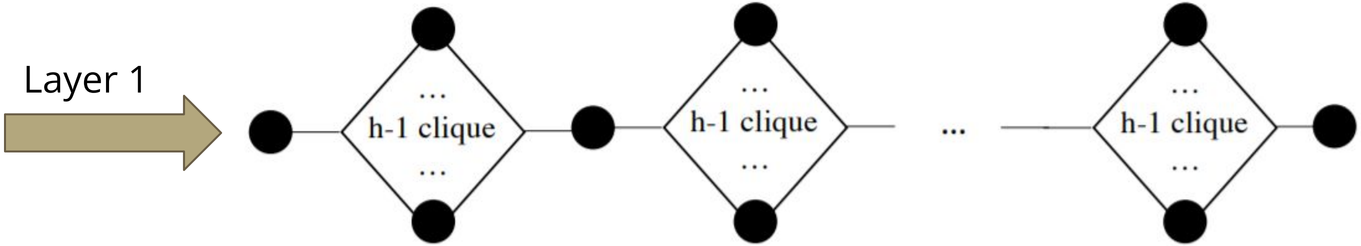


# Trust Graph: Post Processing Algorithm

- Post Processing Goal: Set an **upper bound** on the diameter
- Why? Large diameter (d) trades off with efficiency
- Previous work has shown an upper bound of 2n/h is satisfactory
- Two important adaptations
    - **Sleepy User** adaptation
    - **Unknown h** adaptation

# Trust Graph: Post Processing Algorithm

- Layer k (S_k): Set of all nodes a distance of k away from the “origin”



Layer 1

Figure 1: A multi-layer graph with the layer size alternating between 1 and $h - 1$. Each layer is completely connected within itself.

# Trust Graph: Post Processing Algorithm

- Algorithm: Find the minimum value of |S_k| + |S_k+1| and remove all edges in between these two layers.

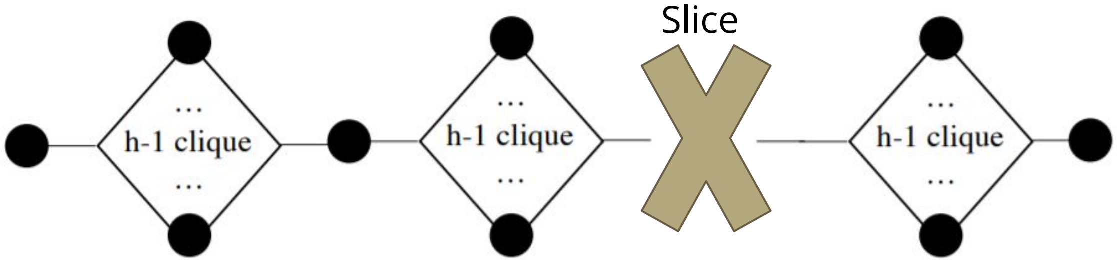


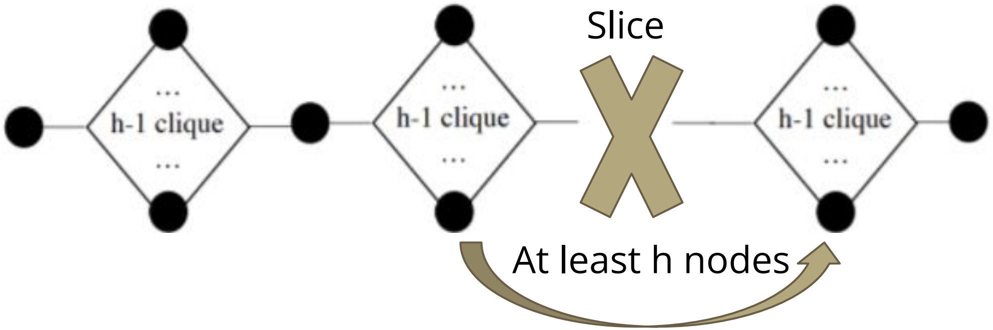
Figure 1: A multi-layer graph with the layer size alternating between 1 and $h-1$. Each layer is completely connected within itself.

# Trust Graph: Post Processing Algorithm

- Two claims to prove
  - Diameter bounded within n/h
  - Never removes edges between honest nodes
- Claim 1 - Diameter bounded
  - Algorithm discards fraction of layers
  - 2/c >= 2n/h

# Trust Graph: Post Processing Algorithm

- Claim 2: Honest nodes remain connected
- Scenario: Corrupt node attempts to remove edges between honest nodes



- Diameter > 2n/h AND 2/c when algorithm applied
- Average sum of two layers → n/(2n/h)*2 = h is greater than the average minimum layer sum

# Building Block 2: Trust Cast

- TrustCast - protocol used to send messages throughout the trust graph
  - New sender S every epoch
  - Epoch = d rounds
  - Verification Function
- Two required results
  - Take action on S
  - No edges removed between honest users

# Knowledge Gaps

- Don’t know n
- Utility of TrustCast
    - Use c to estimate d
- Case 1: Some node k sends to all
- Case 2: Some node k sends to none
- Case 3: Some node k selectively sends

# Consensus Protocol Proper

- Use Trust Graph and Trust Cast → Consensus
- Three phases
  - Happen multiple times until Termination
- Propose Phase
  - Leader selected
  - TrustCasts message
- Vote Phase
  - Vote on a bit
  - heavily impacted
- Commit Phase
  - Commit on a bit

# Vote Phase

- Previous verification function: receive f + 1 votes
- Impossible to receive f + 1 votes
  - Don’t know h
- (1-c)*n + 1 could work but sleepy nodes
- Use the “potentially sleepy” feature of the TrustCast protocol
  - Use (1-c)*k + 1 where k is total online nodes
- Creates a valid condition

# Conclusion

- Successfully adapt to the Dynamic Synchronous Model
- Creating post processing algorithm
- Modified TrustCast and Vote Phase
- Other models to examine
    - Users join in the middle of the protocol
    - Weaker guarantee on starting condition

# Acknowledgements

I would like to acknowledge

- Jun Wan, my mentor, who met with me weekly and answered questions about my research as well as suggested the project
- MIT PRIMES program for giving me the opportunity to research
- Prof. Srini Devadas and all other members of the CS division who organized the conference and the general research of the students

# Thanks for Listening