Byzantine Broadcast with Dishonest Majority

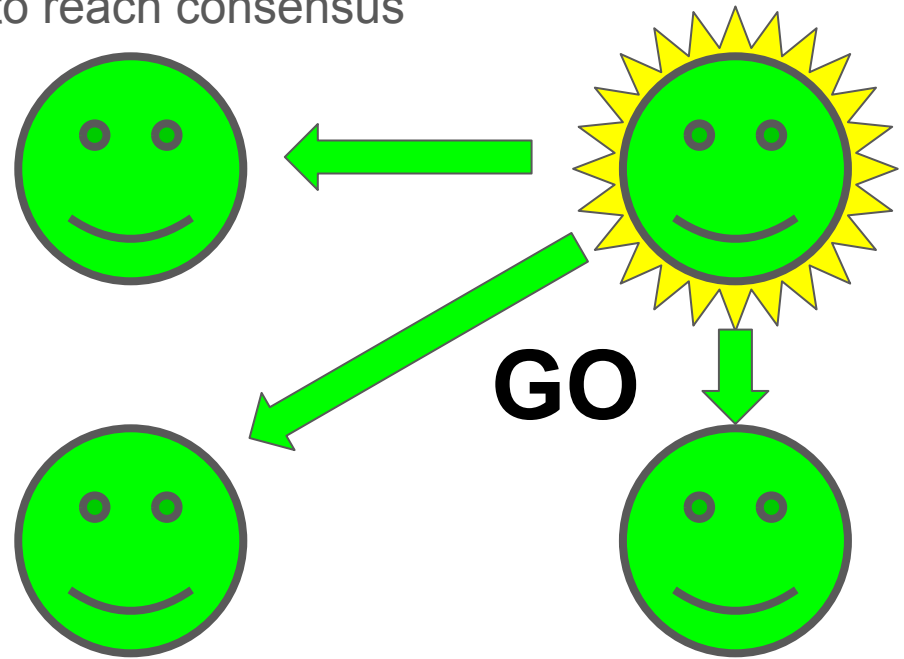
Ezra Gordon under Jun Wan

PROBLEM

Byzantine Broadcast Background→IDEAL

Byzantine Generals are trying to agree on whether to go forward or retreat

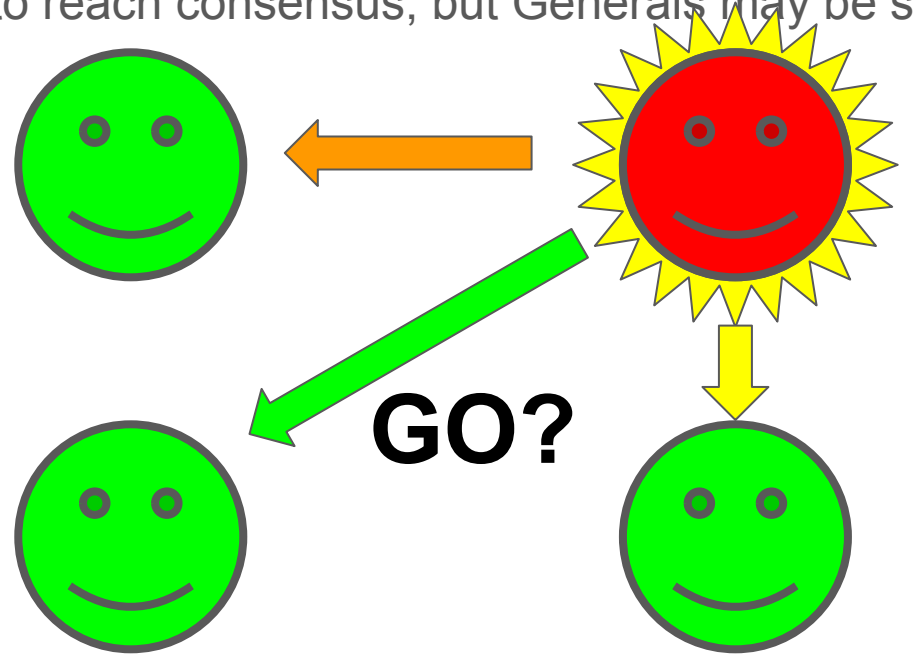
They need a way to reach consensus



Byzantine Broadcast Background→FLAWED

Byzantine Generals are trying to agree on whether to go forward or retreat

They need a way to reach consensus, but Generals may be spies or corrupted

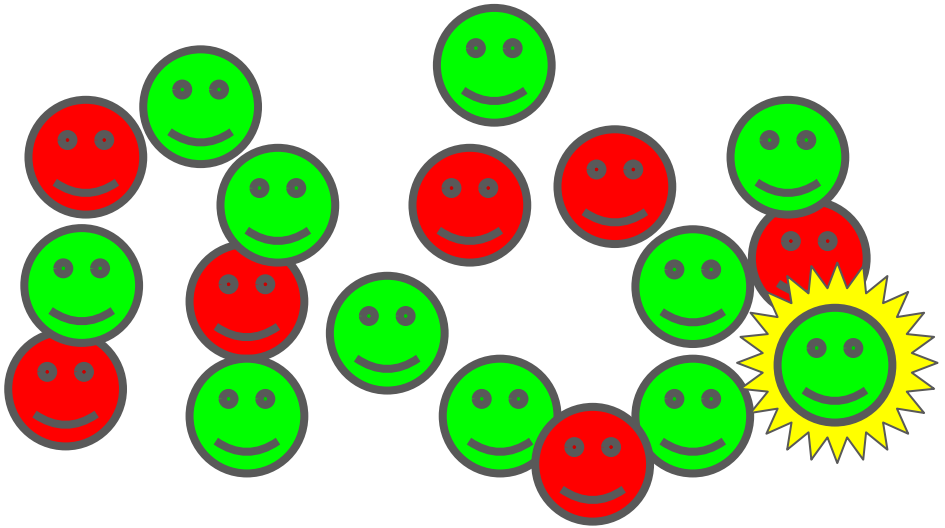


Byzantine Broadcast Background→COMPLICATED

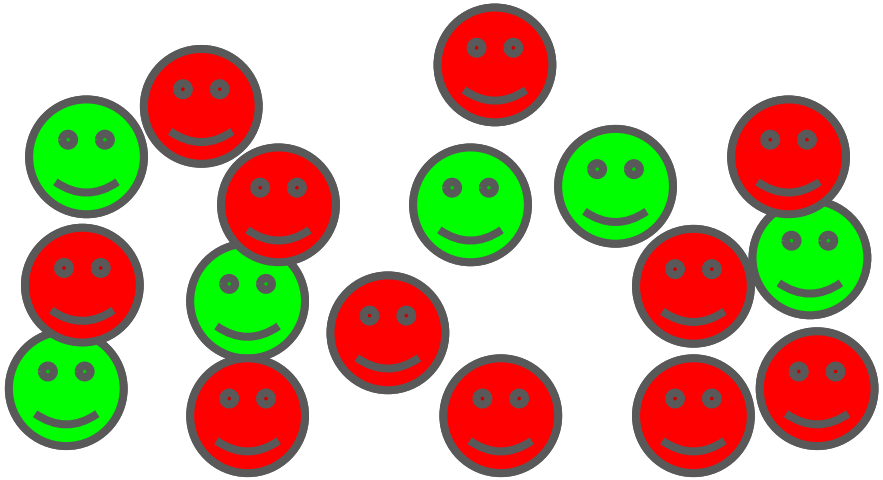
What is a “leader”? → Random, origin of message

They need a way to reach consensus, but Generals may be spies or corrupted

11 Honest
7 Corrupt



Byzantine Broadcast Background→COMPLICATED



11 Corrupt
6 Honest

} Majority is meaningless

Formal Problem Statement:

Given...

1. Honest users all output a message if the leader is honest (termination)
2. Honest users never output different messages (consistency)

SOLUTION

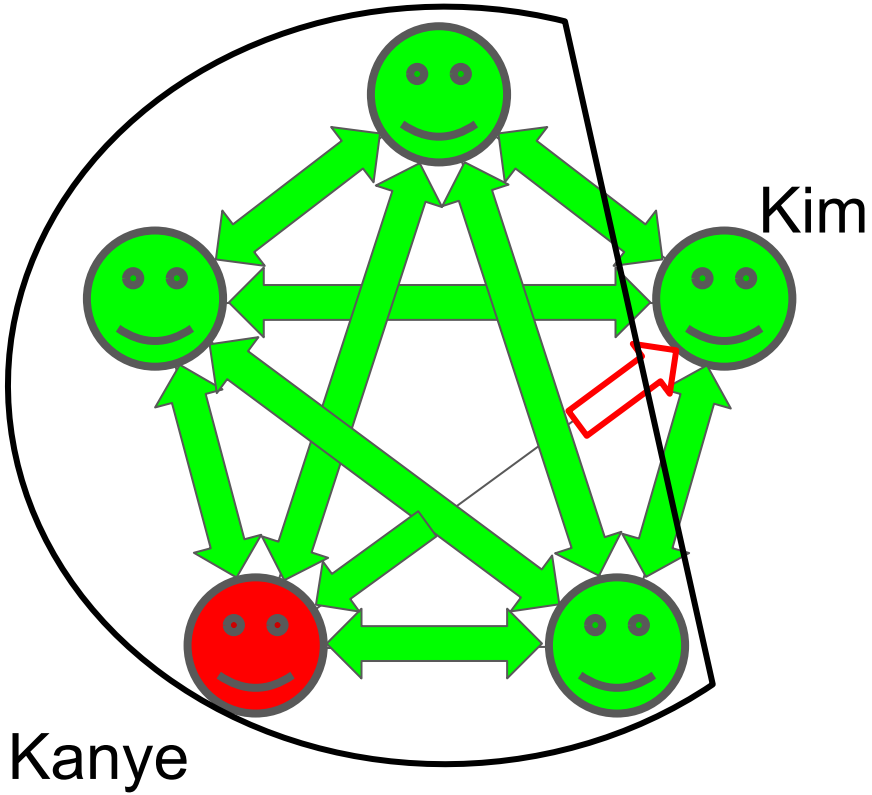
Key Concept: Trust Graphs

Users record
who thinks who is corrupted

Honest users stay
connected

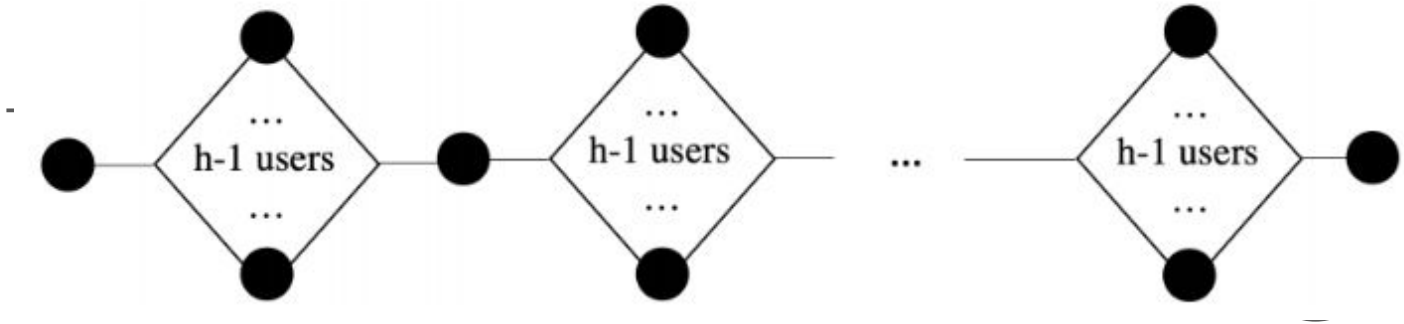
Trust graphs are distinct

4 Honest
1 Corrupt



What's the Point of Keeping Trust Graphs?

- Gives a way to remove/ignore corrupt users:
  - Within *x* rounds of communicating, users always receive messages from other users that are



irements

Trust graph diameter upper bound (d)=

$$\left\lfloor \frac{USERS}{HONEST\ \ USERS} \right\rfloor + \left\lceil \frac{USERS}{HONEST\ \ USERS} \right\rceil - 1$$

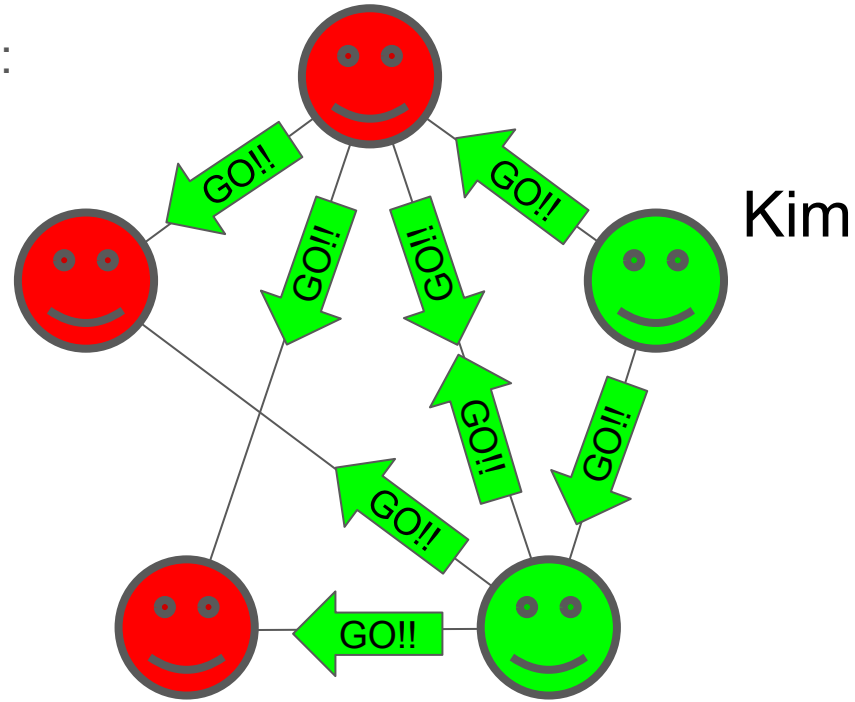
Key Concept: The Gossip Function

How each part of our protocol operates:

Gossip(*sender, message, rounds)*

ex:

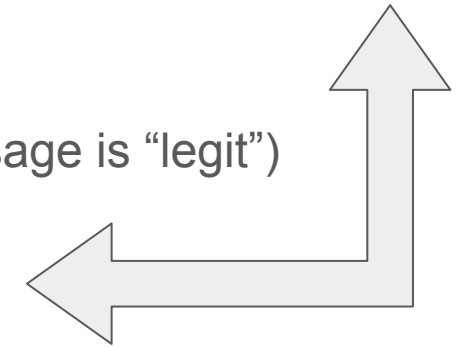
Gossip(*Kim*, “*GO!!*”, 2*)*



Intuition of Solution

- Three Step protocol:

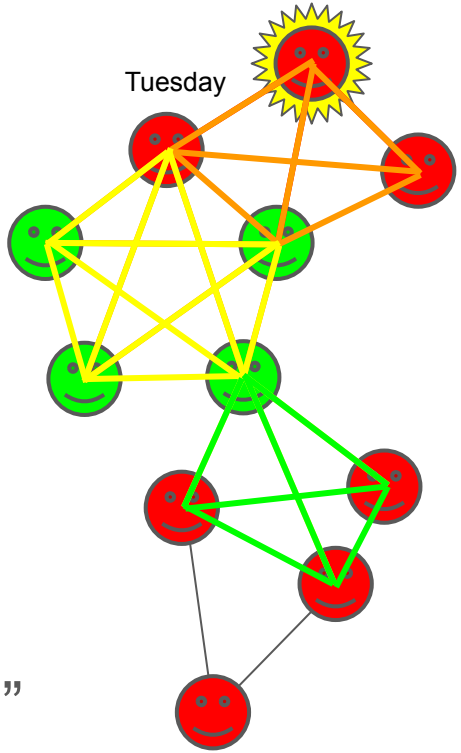
    1. The leader broadcasts a message, users then **RELAY** messages sent by the leader
    2. Users “**VOTE**” on what to do (whether the message is “legit”)
    3. Users decide/share their choice to **COMMIT**



- Most users are corrupt, so the steps become more drawn out

Relay Step

Gossip(Leader, $message_{Leader}$, d)



Why:

- So every user has something to vote on
- So users know if the leader “equivocated”

Vote Step

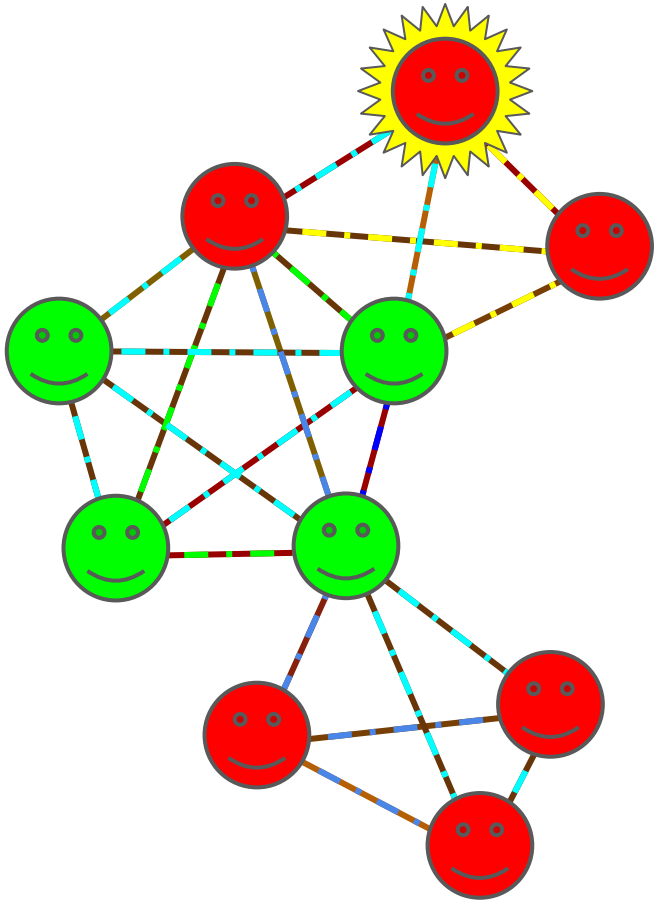
Gossip(Every user i, V_i, d)

and... when i receives V_j: Gossip(i, confirm-V_j, d)

Why:

So every user knows what everyone plans to do

So every user has a record of other users receiving votes



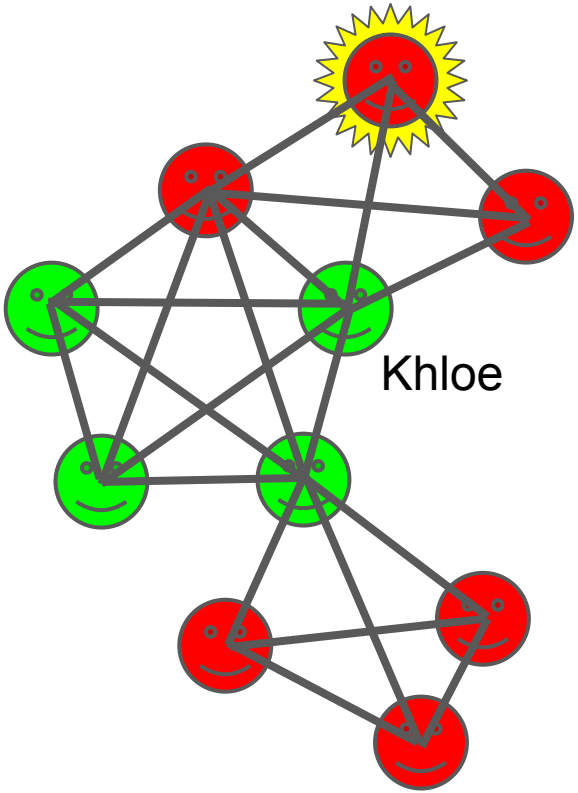
Commit Step

Gossip(Every user *i*, “*commit*”, *d*)

if...

Why:

So users receive confirmation that they should “terminate”



Termination

Users are carefully instructed such that an honest/flawless leader cannot be undermined:

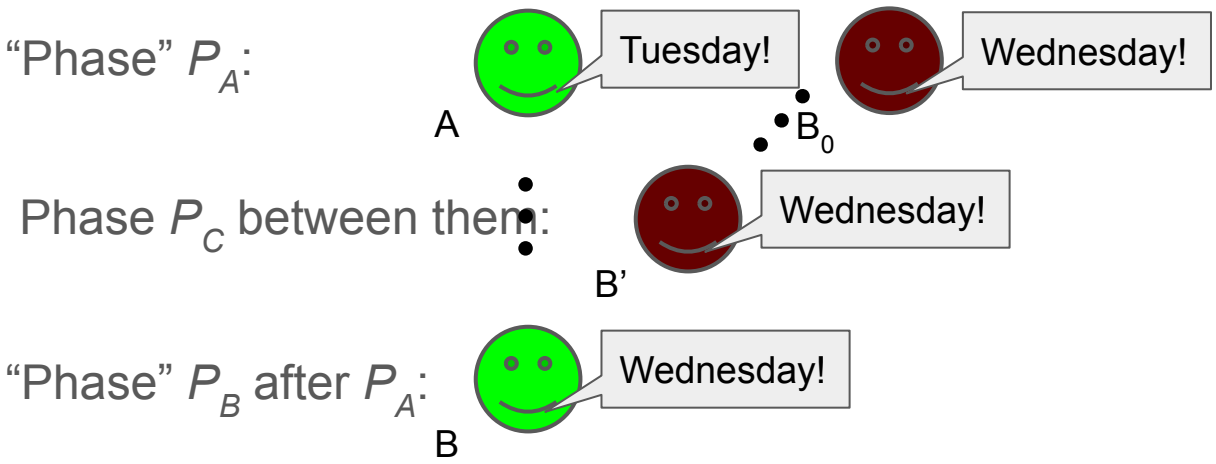
- Malicious users cannot impersonate or frame the leader

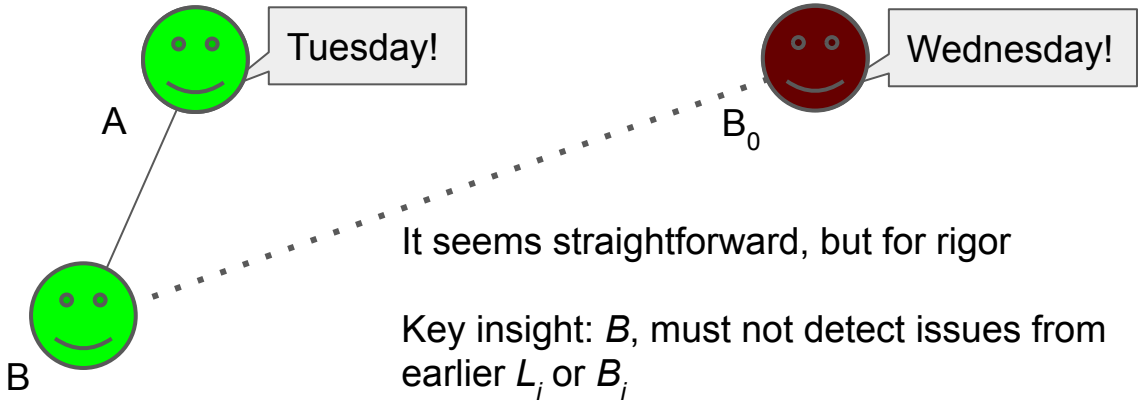
- Protocol dictates that malicious users must act honest or be removed

Consistency - Why “vote” for 2d rounds?

Same round consistency → Voting detects issues

Different round consistency → More complicated





It seems straightforward, but for rigor

Key insight: *B*, must not detect issues from earlier L_i or B_i

If *B* trusts B_0, *B* must have received "Wednesday" before *A* committed to Tuesday, and sent it to *A*, contradicting the fact that *A* committed

This only occurs if:

Users can claim they didn't receive sufficient information to not commit

Thank you!

Special thanks to:

Jun Wan

Professor Devadas

Professor Gerovitch

PRIMES Program

MIT

Abstract

Byzantine broadcast is a well-studied consensus-building problem in computer science. A randomly chosen leader must ensure all honest users agree on the same message. Broadly speaking, most literature/results for this problem rely on an honest majority of users in the protocol. For this project, worked to improve and simplify his existing protocol and proof for with sub-linear round complexity under a dishonest majority of users. We also explored proofs for theoretical minimum round complexity under a dishonest majority.

Thoughts on organization

-1-3 (more) slides on general byzantine agreement

-1 slide on specific parameters for us

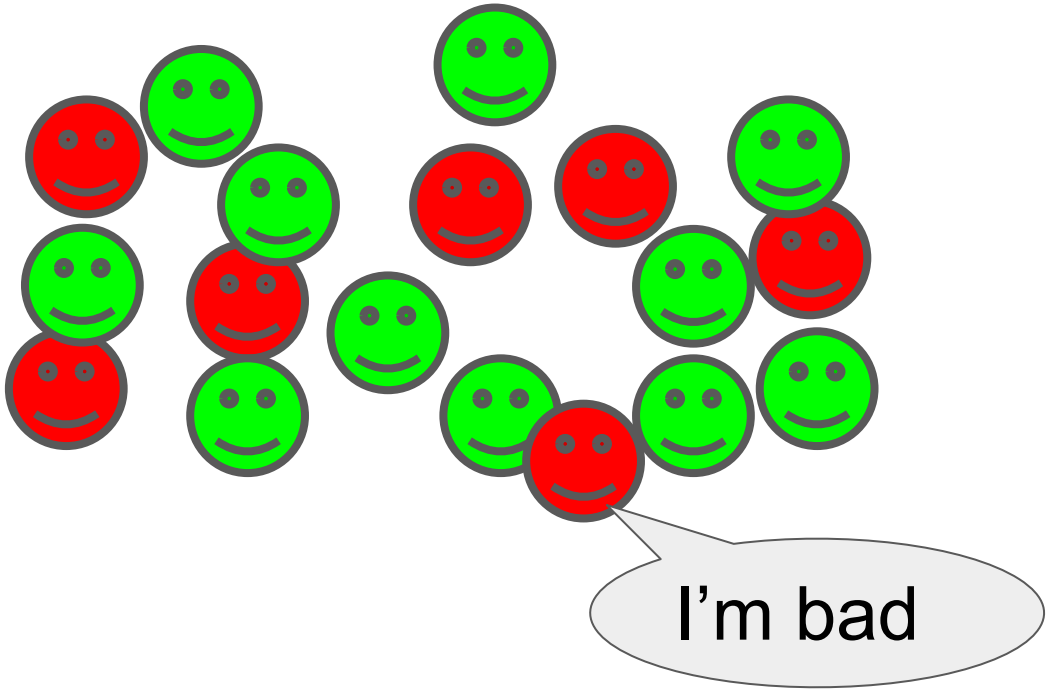
-1-2 slides on trust graphs (maybe another for equivocation)

-2-3 slides explaining the protocol

-2-3 slides outlining the proof

-There is probably something else too

Byzantine Broadcast Background→COMPLICATED



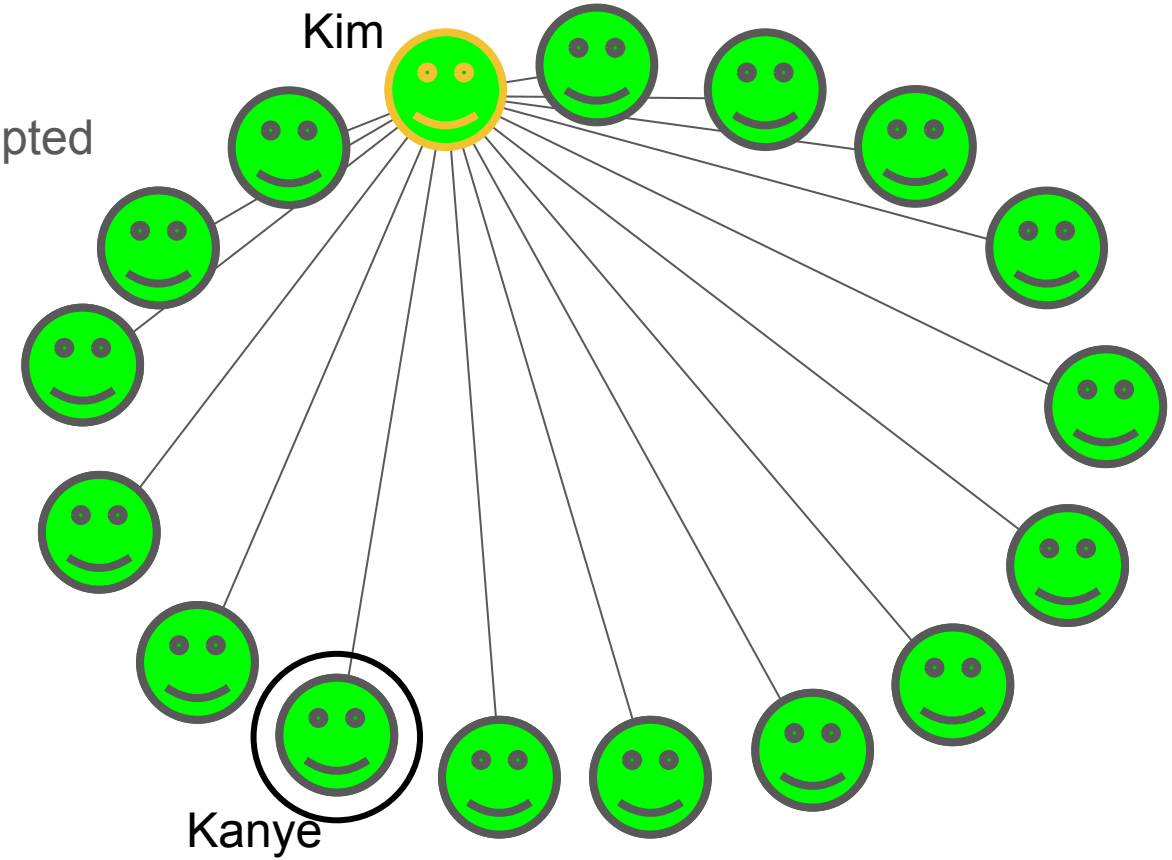
11 Honest
6 Corrupt

Key Concept: Trust Graphs

Users record
who thinks who is corrupted

Users need to be
connected to 5 others

11 Corrupt
6 Honest



Revisited Solution

“Equivocation” &
Users have something to vote on

Three Step protocol:

1. A leader broadcasts a message, users then **RELAY** messages sent by the leader (d rounds)
2. Users “**VOTE**” on what to do (2*d rounds)
3. Users decide/share their choice to **COMMIT** (d rounds)

Assuring common knowledge &
Preventing later disagreements

Announcing commitment

Parameters

- In different rounds, users send “signed” messages to one another. (Signatures can’t be faked)
- Users initially always send updates to everyone
- User X outs themself as malicious to user Y if:
  - X doesn’t send a message to Y
  - X sends two messages that conflict*
  - X otherwise doesn’t follow instructions...
- Users record who “trusts” who in a “trust graph”