# Compute Globally, Act Locally: Protecting Federated Systems from Systemic Threats


Arjun Narayan

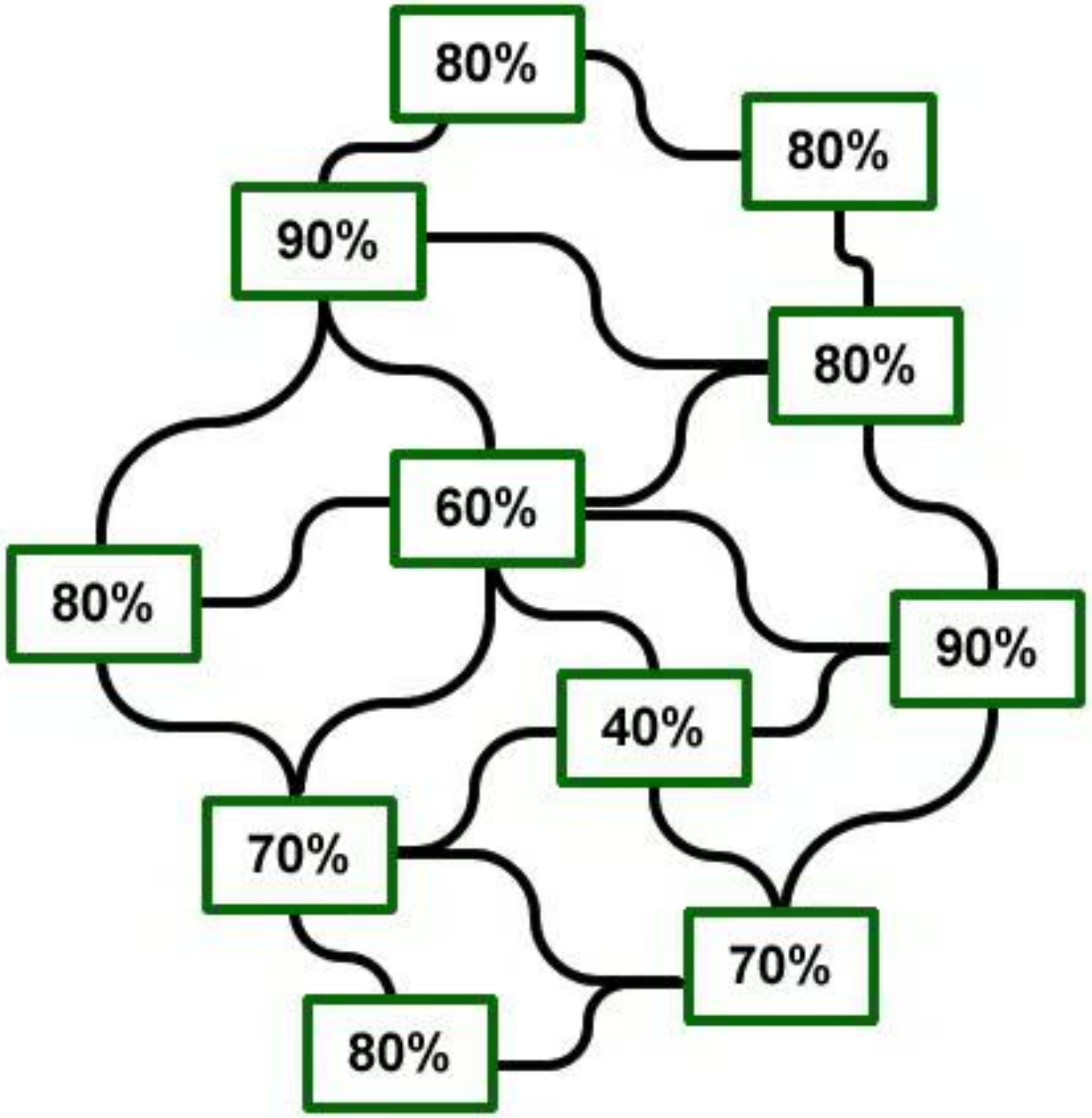
Antonis Papadimitriou

Andreas Haeberlen

University of Pennsylvania


# Motivation

- Interdependent systems are vulnerable to **cascading failures**.
  - Routing
  - Load balancing
- Solving this often requires a **global view**.
- This is a well known fact in the distributed systems world.
- This insight can be generalized.



# Motivation

Remember the 2008 Financial Crisis?

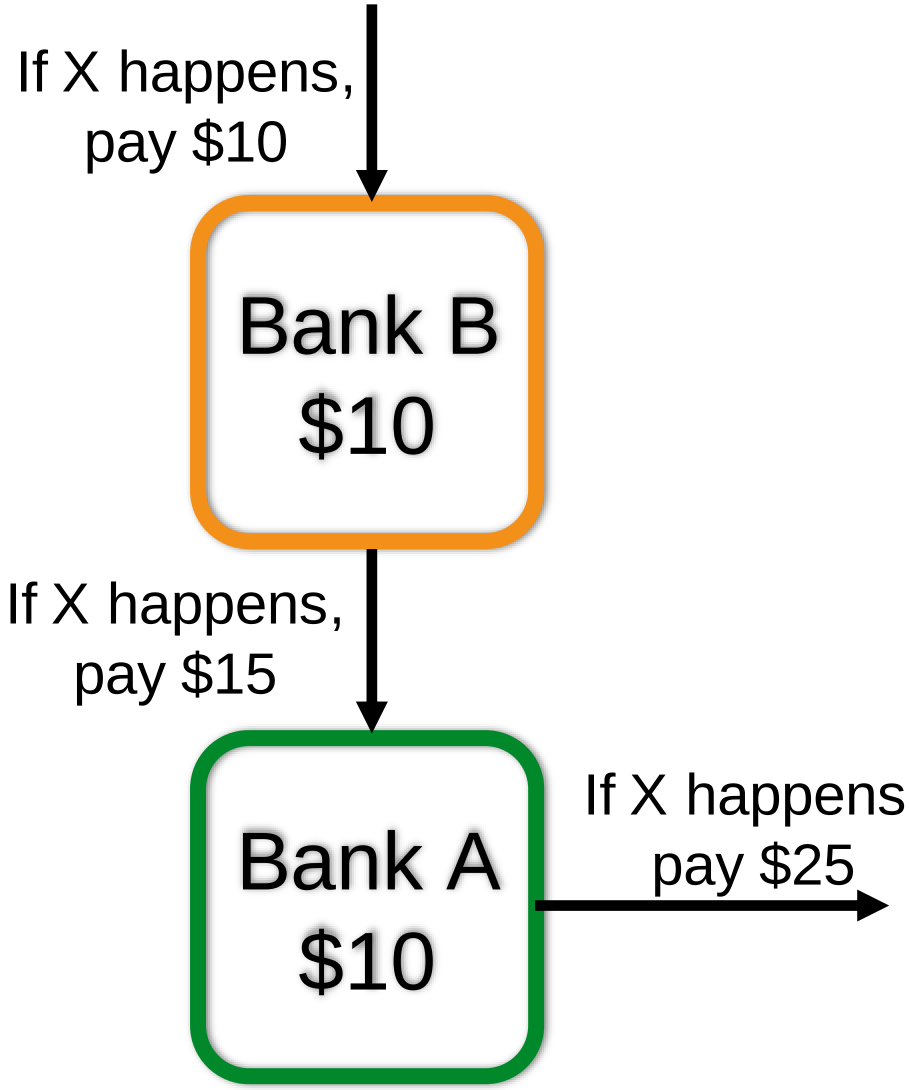
Why did nobody see it coming?

There was no global view.

Let me start with some background on banking.

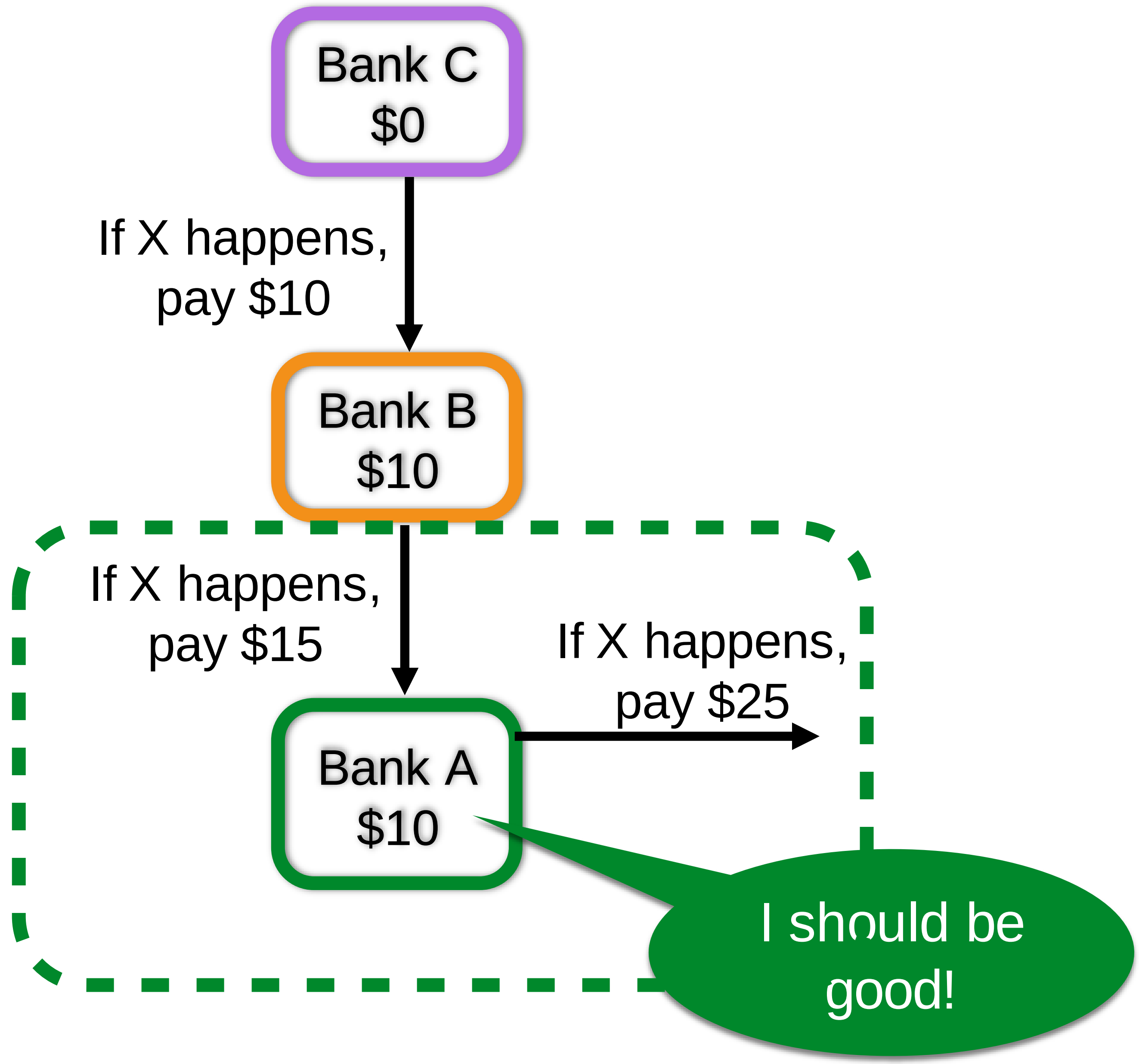
# What is Systemic Risk?

- Banks have some liquid reserves.
- A bank gains exposure to risk as part of its normal business. We can model these as hypothetical events.
- Banks want their **net** risk to be contained
- They offload surplus risk to other banks
- This creates a network of dependencies.



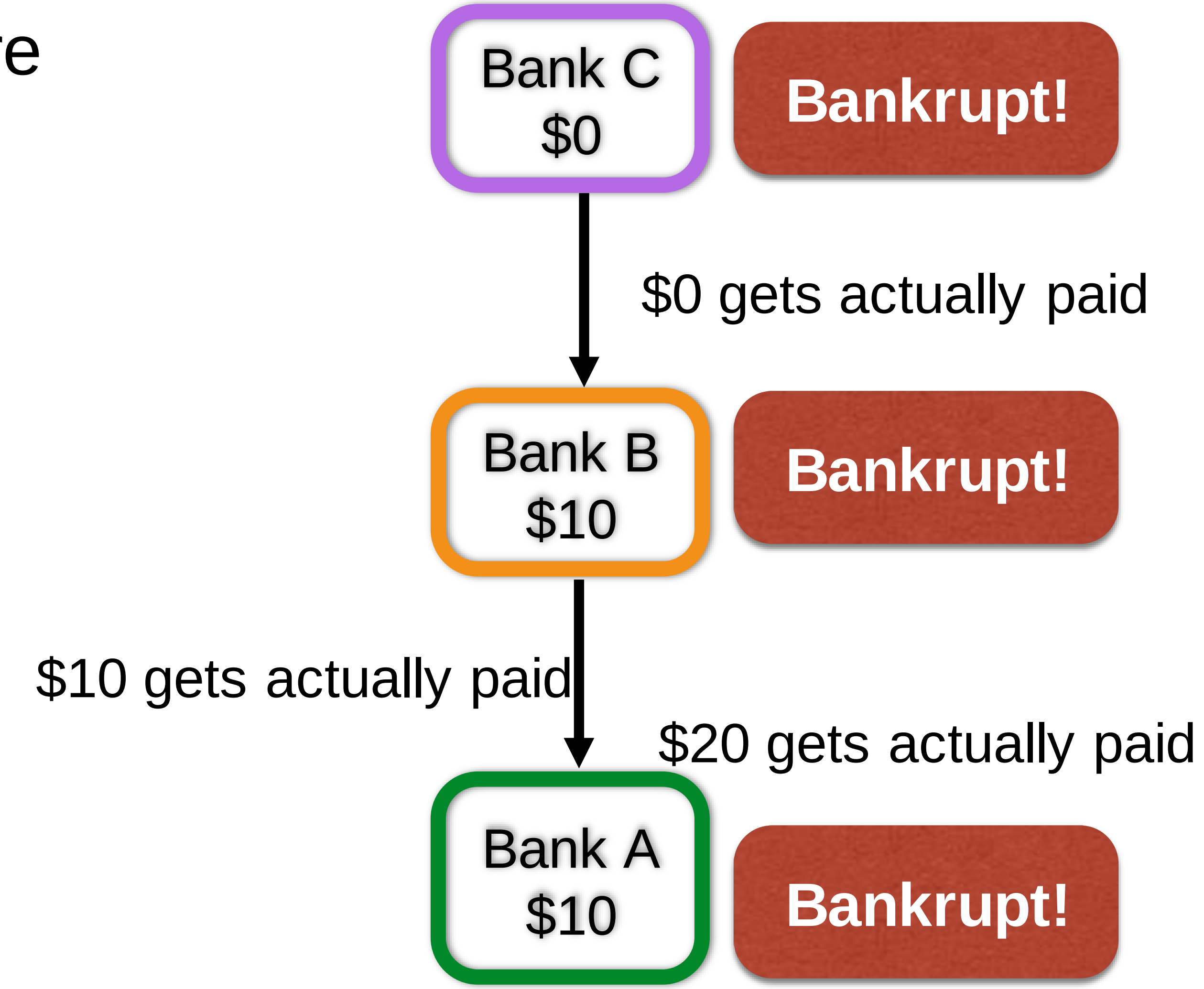
# What could go wrong?

- Banks only have a local view
- So their local conclusions are vulnerable to **counterparty risk**



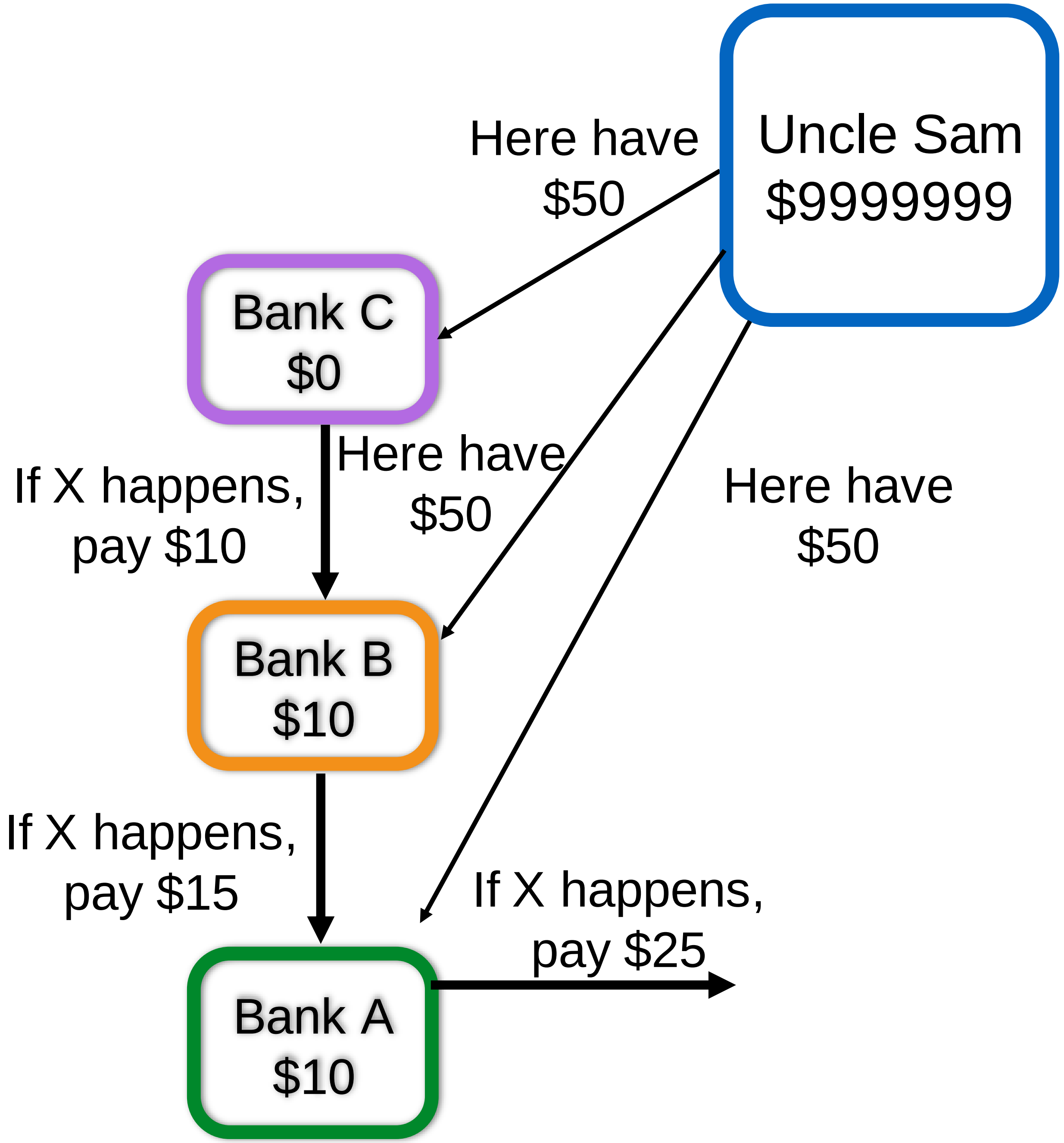
# What could go wrong?

- Banks only have a local view
- So their local conclusions are vulnerable to **counterparty risk**
- Consider another upstream bank C that is faulty
- What happens?



# What Now?

- This uncertainty creates a financial panic.
- But there is a solution!
- (Nobody likes that solution…)
- Is there another way?

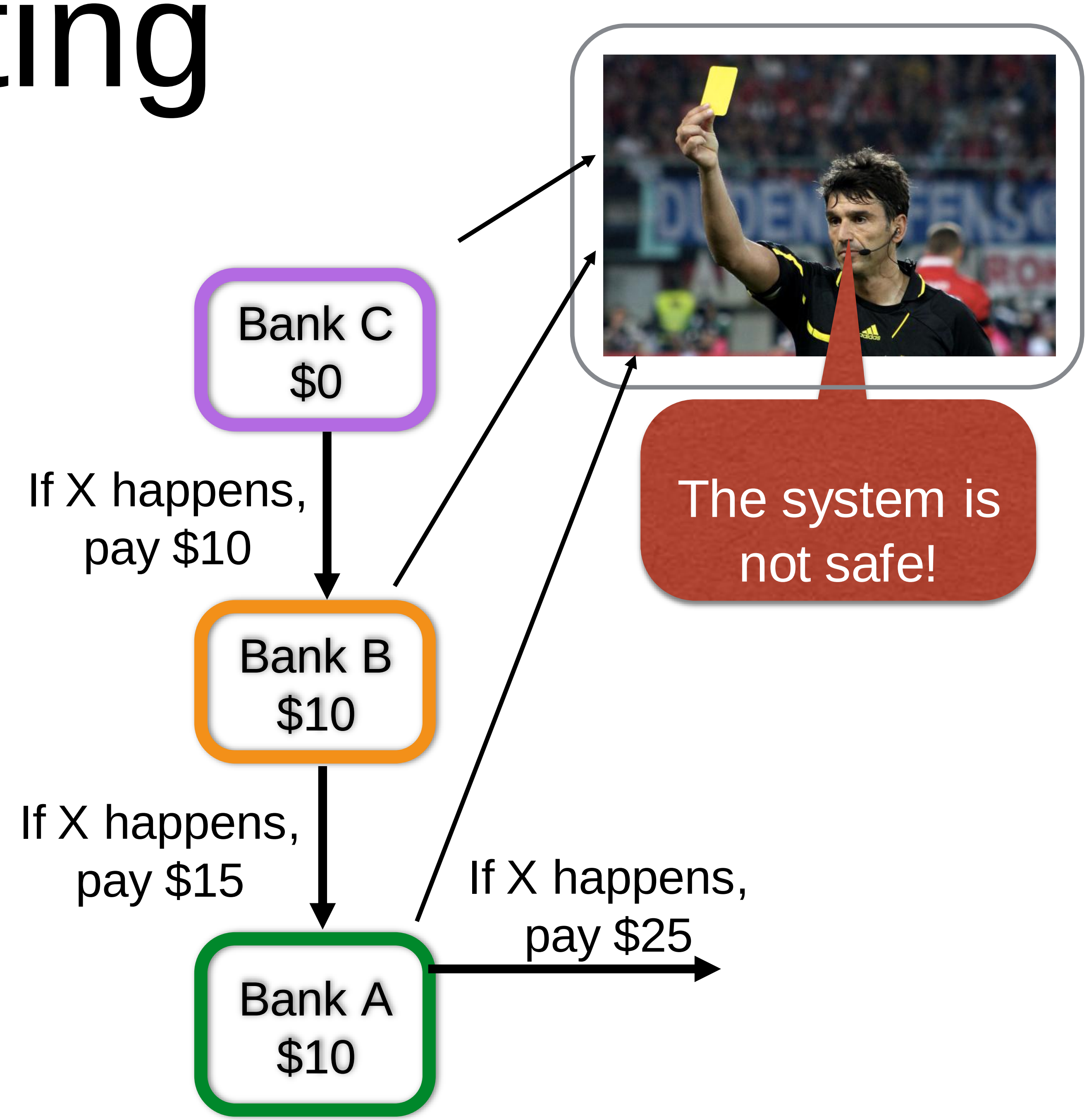


# How can we prevent this?

- We need an **early warning system** to measure systemic risk.
- Today we do individual bank-level stress tests.
  - But as we have seen, this is insufficient.
- We need a more comprehensive system that would:
  - Take information from every bank,
  - Compute **global** checks,
  - Output this to regulators.

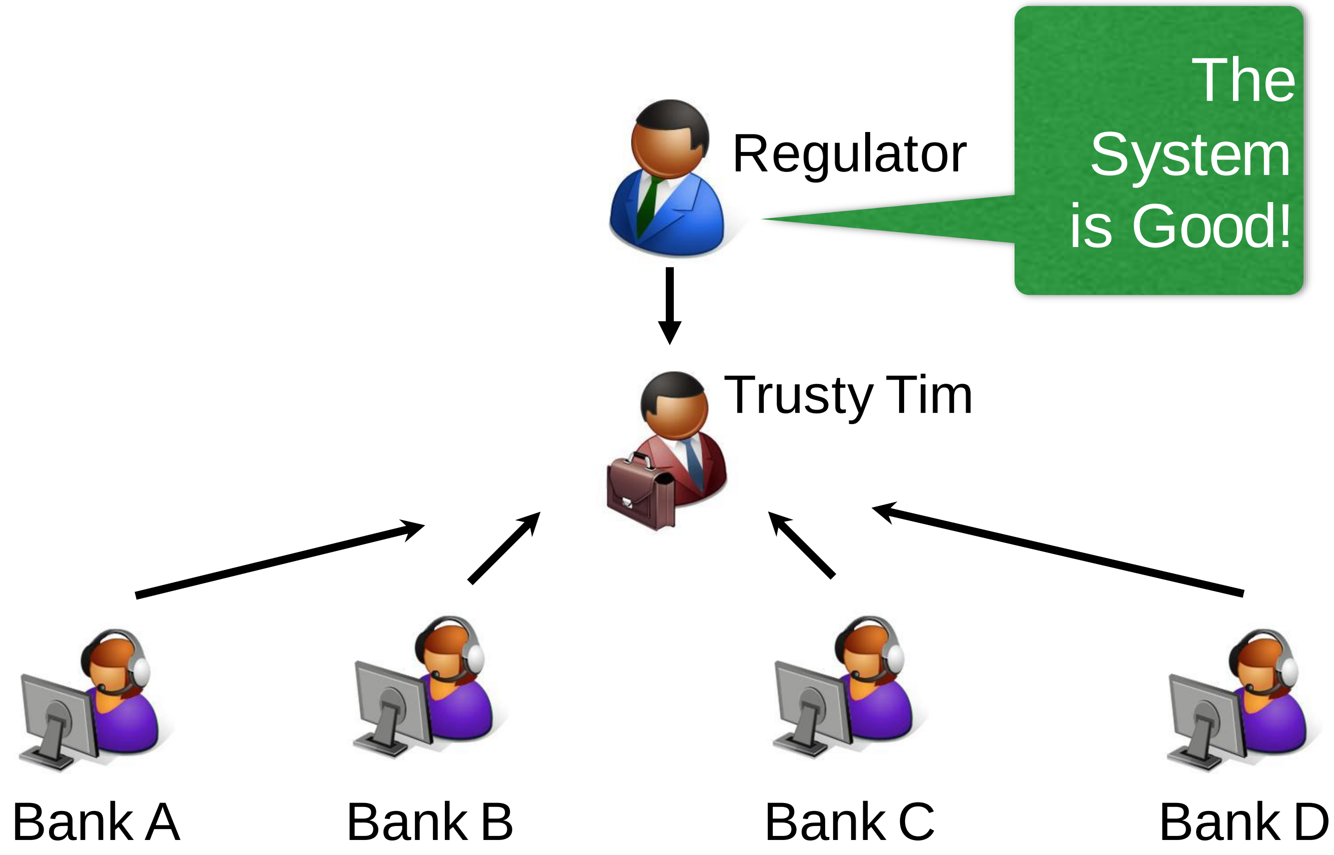
# System Wide Stress Testing

- What would a test compute?
- We are not economists.
- However, economists have thought about this question!
- Models exist.
- They know **what** to compute…
- … but they don't know **how**.



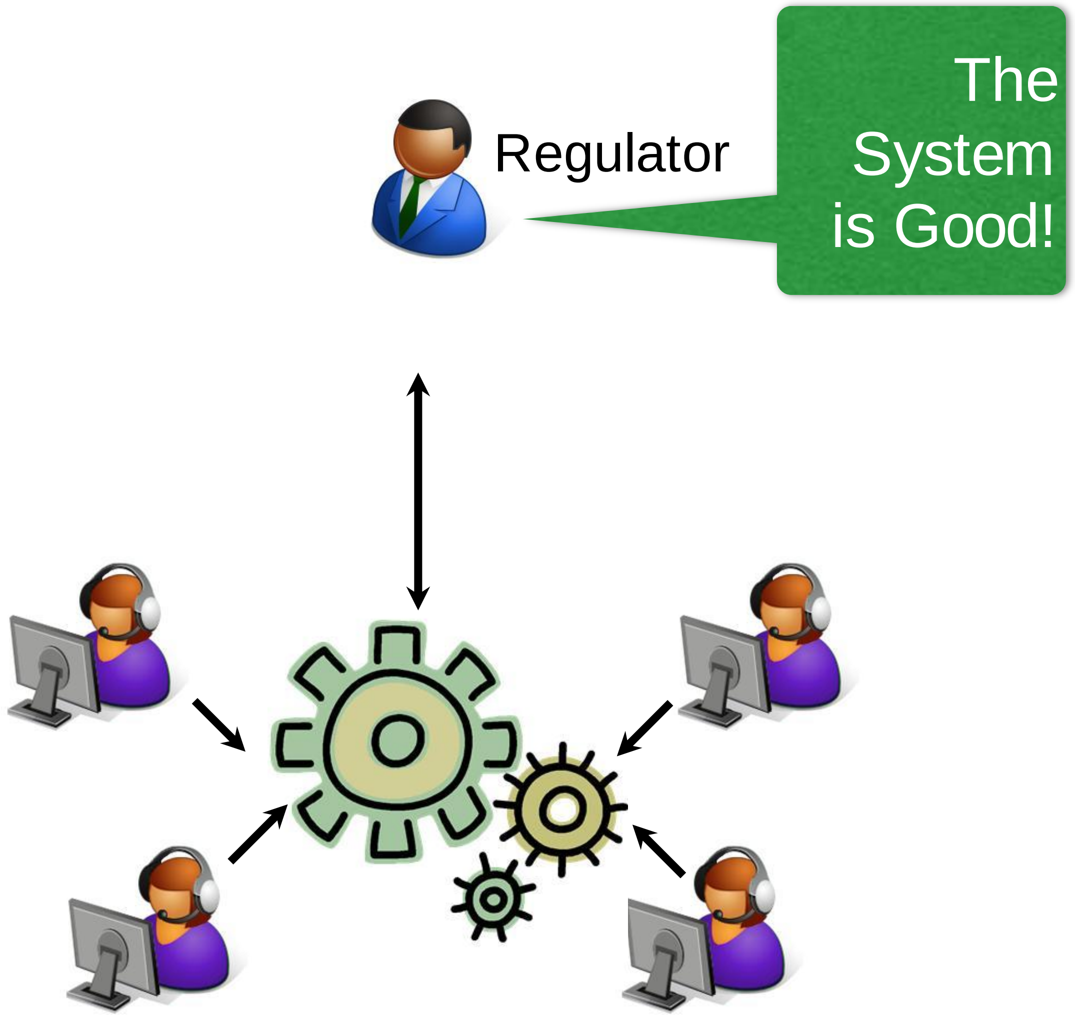
# System Wide Stress Testing

- How do we conduct **systemic** stress tests?
- Idea: Give all the data to a central regulator.
- Doesn't work, because that is too much power for one party.



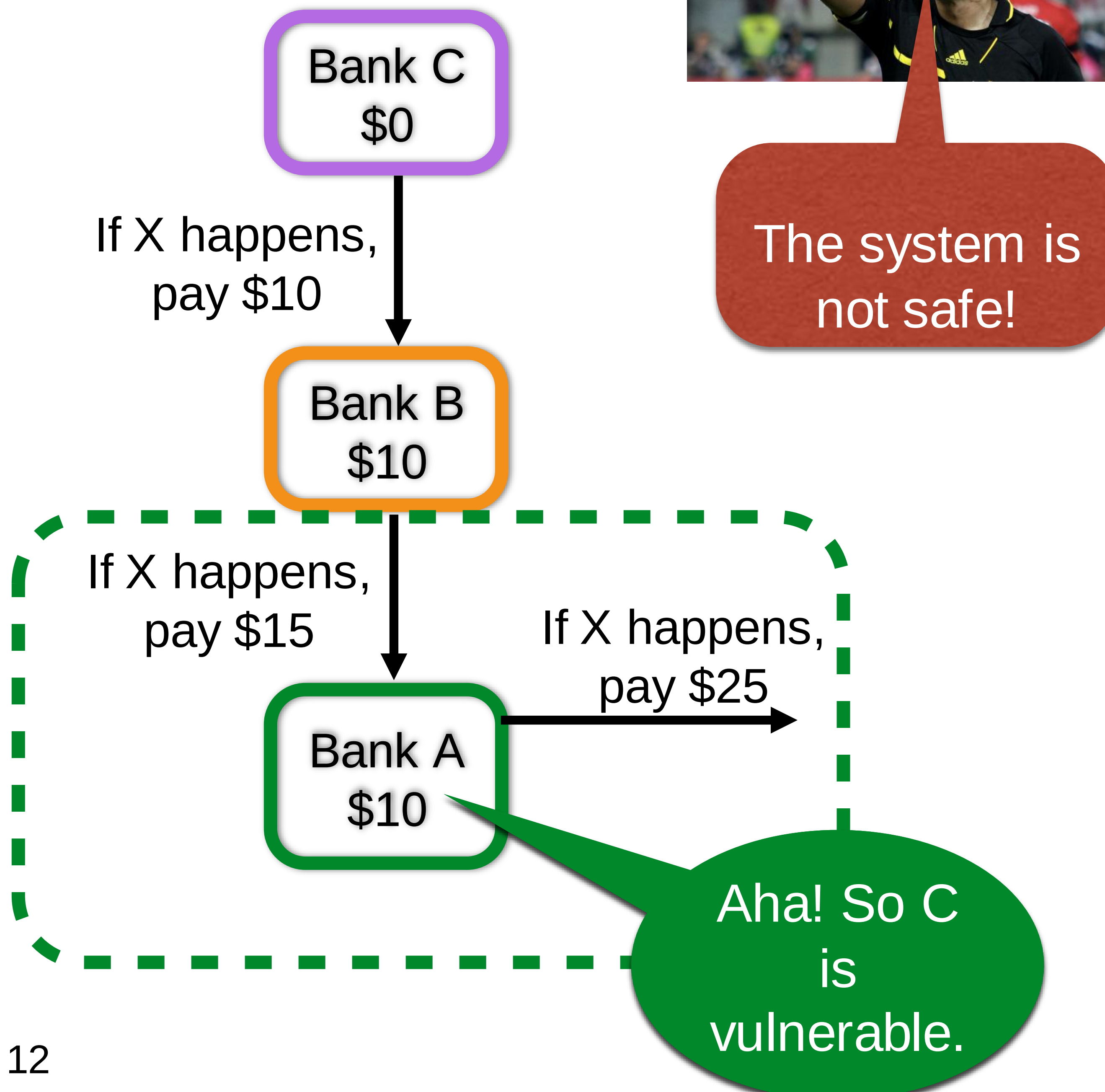
# System Wide Stress Testing

- How do we conduct **systemic** stress tests?
- Idea: Give all the data to a central regulator.
- Doesn't work, because that is too much power for one party.
- Idea: Use Secure Multiparty Computation (MPC).
- This doesn't scale.
- Is still not necessarily private.



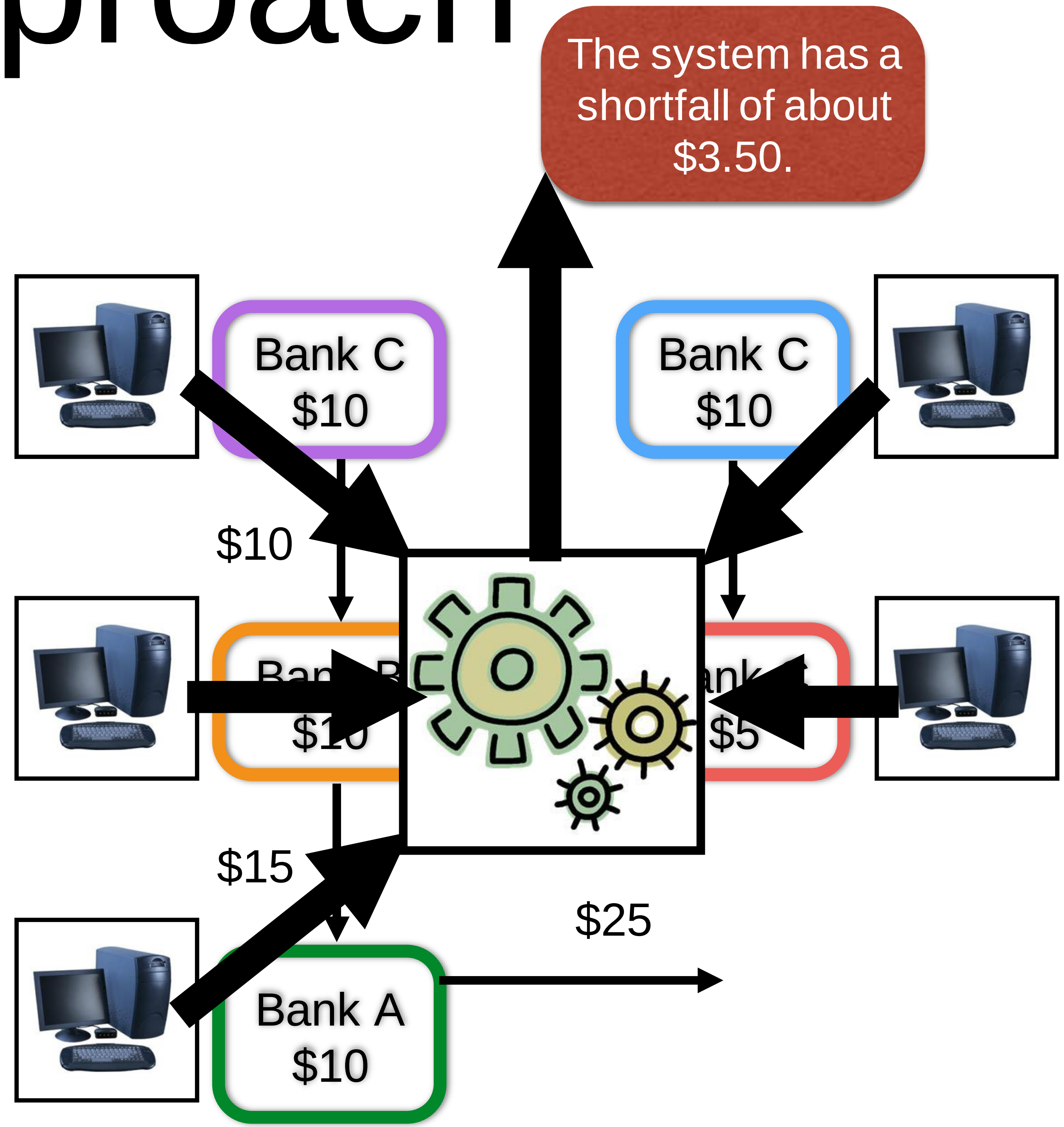
# Building an Early Warning System

- We want to build a **distributed system** that tells us if the system as a whole is risky.
- Challenge 1: **Privacy**
- The output of the computation should protect the banks' proprietary information.
- Challenge 2: **Scalability**
- The system should be scalable to hundreds of banks.



# Our Approach

- Each bank has an associated node.
- The nodes run a series of multiparty computations.
- We can exploit the fact that these algorithms are graph algorithms with limited degree.
- The output of the computation is **differentially private**.
- So how do we do this?

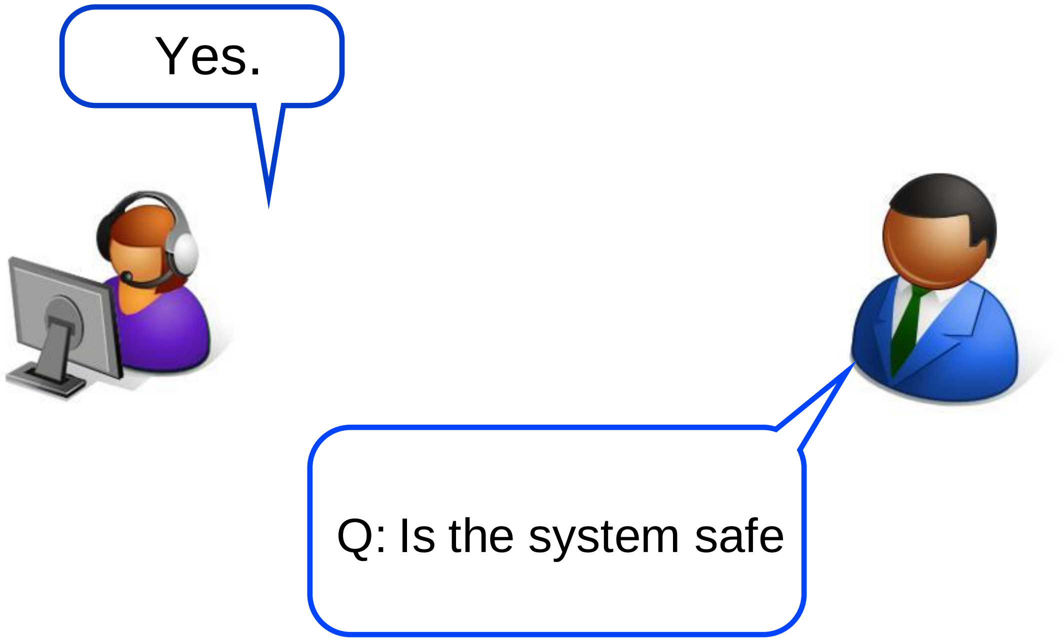


# Outline

- Motivation
- The Case for Systemic Stress Testing
- Building an Early Warning System
- Background:
  Differential Privacy
  Economic Models
- Our Approach:
  Limited MPC
  Secret Sharing
- Status

# Background: Differential Privacy

- Provides **provable** privacy guarantees. (Dwork, Nissim, McSherry, Smith 2006)
- Protects against **auxiliary information** attacks.
  - This is very important!
  - Netflix deanonymization.
  - AOL deanonymization.
- This is hard to reason about!

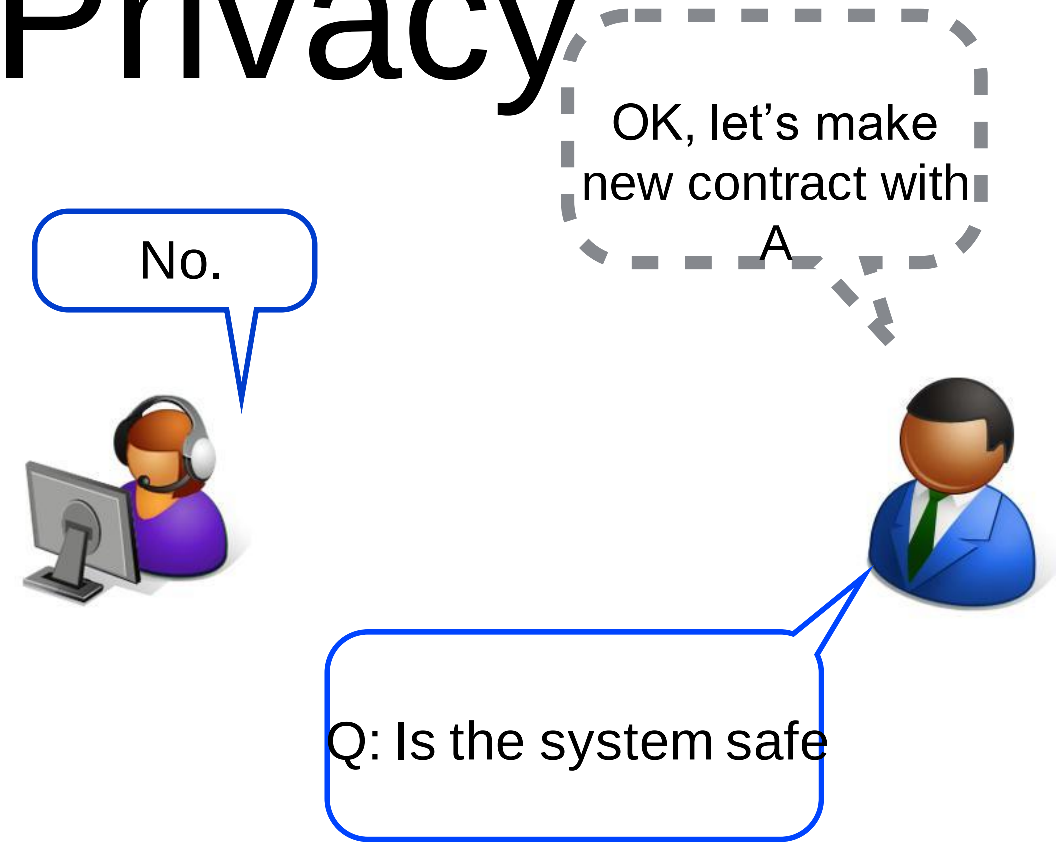


# Background: Differential Privacy

- Provides **provable** privacy guarantees. (Dwork, Nissim, McSherry, Smith 2006)
- Protects against **auxiliary information** attacks.
  - This is very important!
  - Netflix deanonymization.
  - AOL deanonymization.
- This is hard to reason about!

# Background: Differential Privacy

- Provides **provable** privacy guarantees. (Dwork, Nissim, McSherry, Smith 2006)
- Protects against **auxiliary information** attacks.
  - This is very important!
  - Netflix deanonymization.
  - AOL deanonymization.
- This is hard to reason about!



# Background: Differential Privacy

- Provides **provable** privacy guarantees. (Dwork, Nissim, McSherry, Smith 2006)
- Protects against **auxiliary information** attacks.
  - This is very important!
  - Netflix deanonymization.
  - AOL deanonymization.
- This is hard to reason about!

# Background: Differential Privacy

- Provides **provable** privacy guarantees. (Dwork, Nissim, McSherry, Smith 2006)
- Protects against **auxiliary information** attacks.
- Works by adding a little noise to answers.
  - Noise thwarts adversaries looking to exploit edge cases.
  - What we care about are large effects, so the noise is okay.

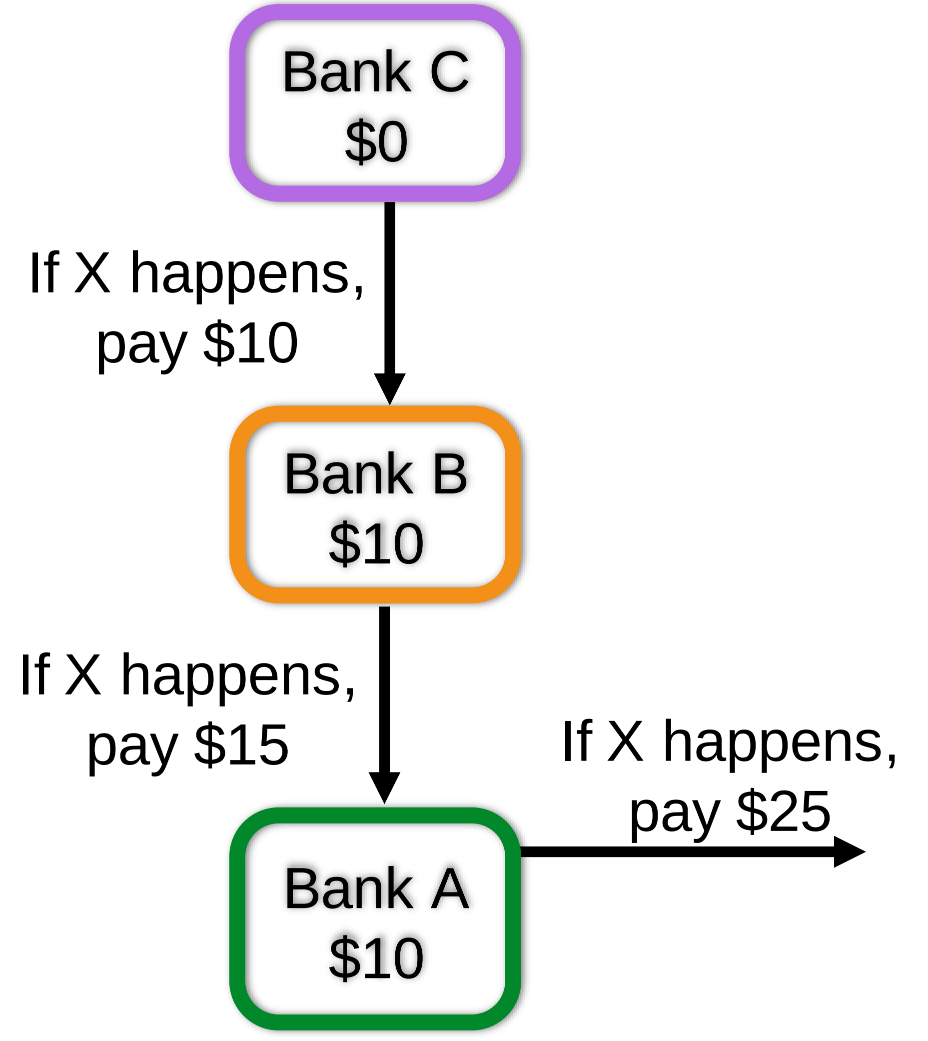
The system is not safe…ish

# Background: The Structure of Economic Models

- There are many economic models of financial crises.
- They roughly have the same structure:
- Simulate “what-if” scenarios on bank connections,
- and compute how much trouble the system is in.

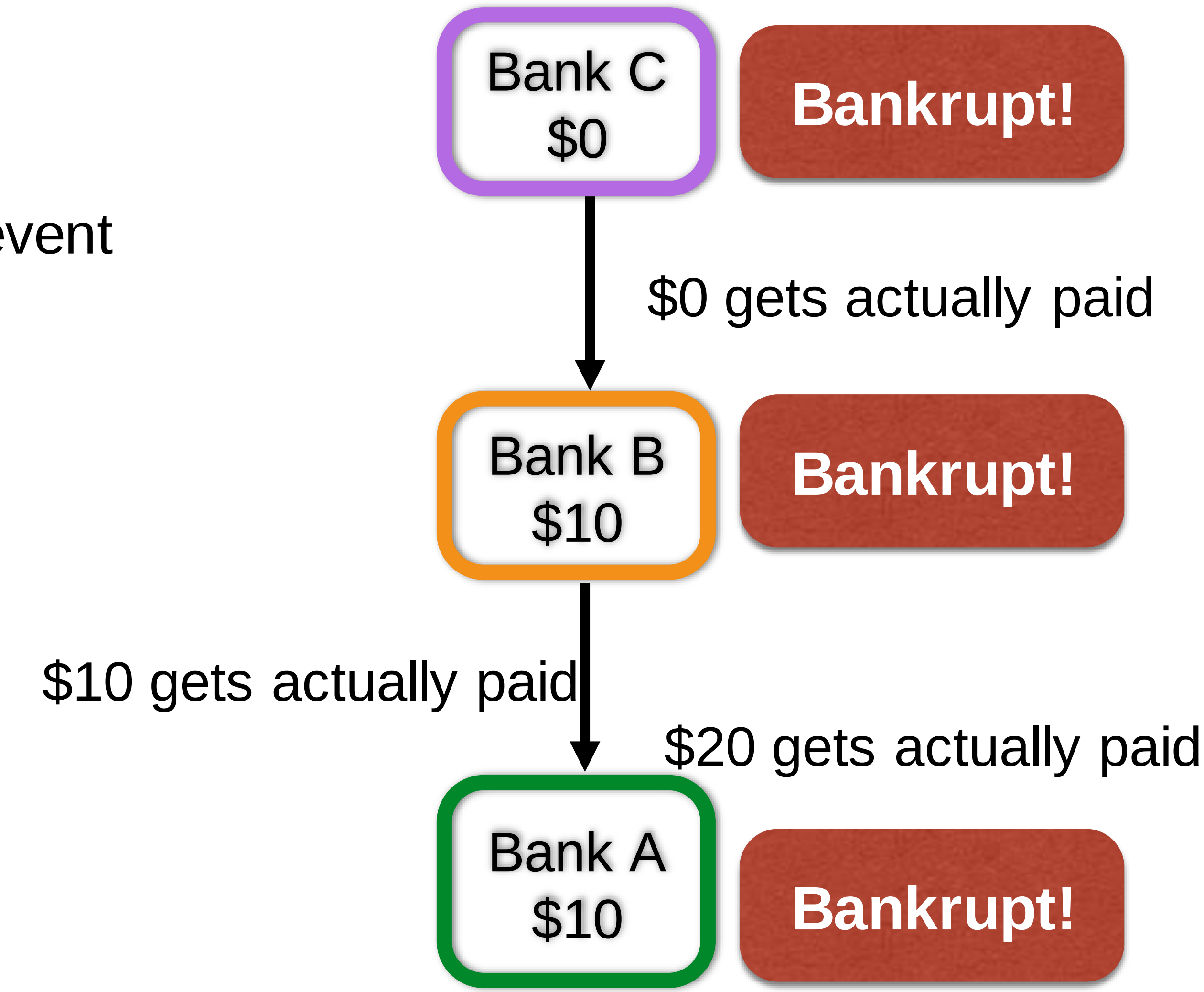
# A Closer Look

- The algorithm I've presented is a simplified version of Eisenberg and Noe, 2001.
- Intuitively what it does is it plays through what would happen if the event were to occur.
- But this is really a graph algorithm:
  Initialization
  Communication
  State Update
  Aggregation
- Nice properties:
  Convergence to unique solution,
  Termination in linear number of iterations.



# A Closer Look

- The algorithm I've presented is a simplified version of Eisenberg and Noe, 2001.
- Intuitively what it does is it plays through what would happen if the event were to occur.
- But this is really a graph algorithm:
  Initialization
  Communication
  State Update
  Aggregation
- Nice properties:
  Convergence to unique solution,
  Termination in linear number of iterations.




3 bankruptcies

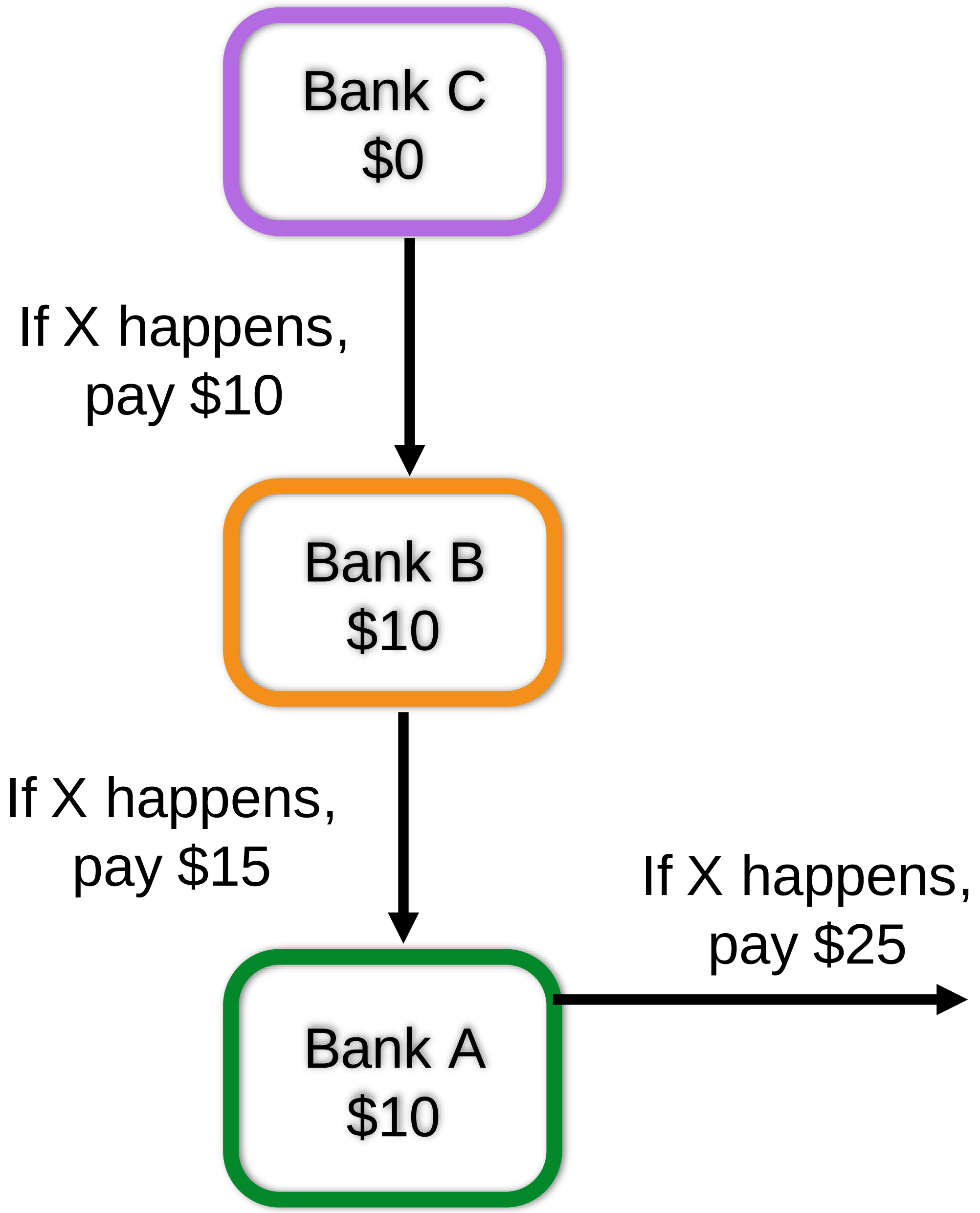
# Computing These Models

- Naively computing matrix multiplications in MPC won't work.
- Just as in PageRank…
- Iterative graph-based approaches are easier to execute…
- Especially when we take advantage of sparsity.

$$\begin{pmatrix} 0 & \$10 & 0 & \$5 \\ 0 & 0 & \$15 & \$10 \\ \$10 & 0 & 0 & 0 \\ 0 & 0 & \$15 & 0 \end{pmatrix}^T$$

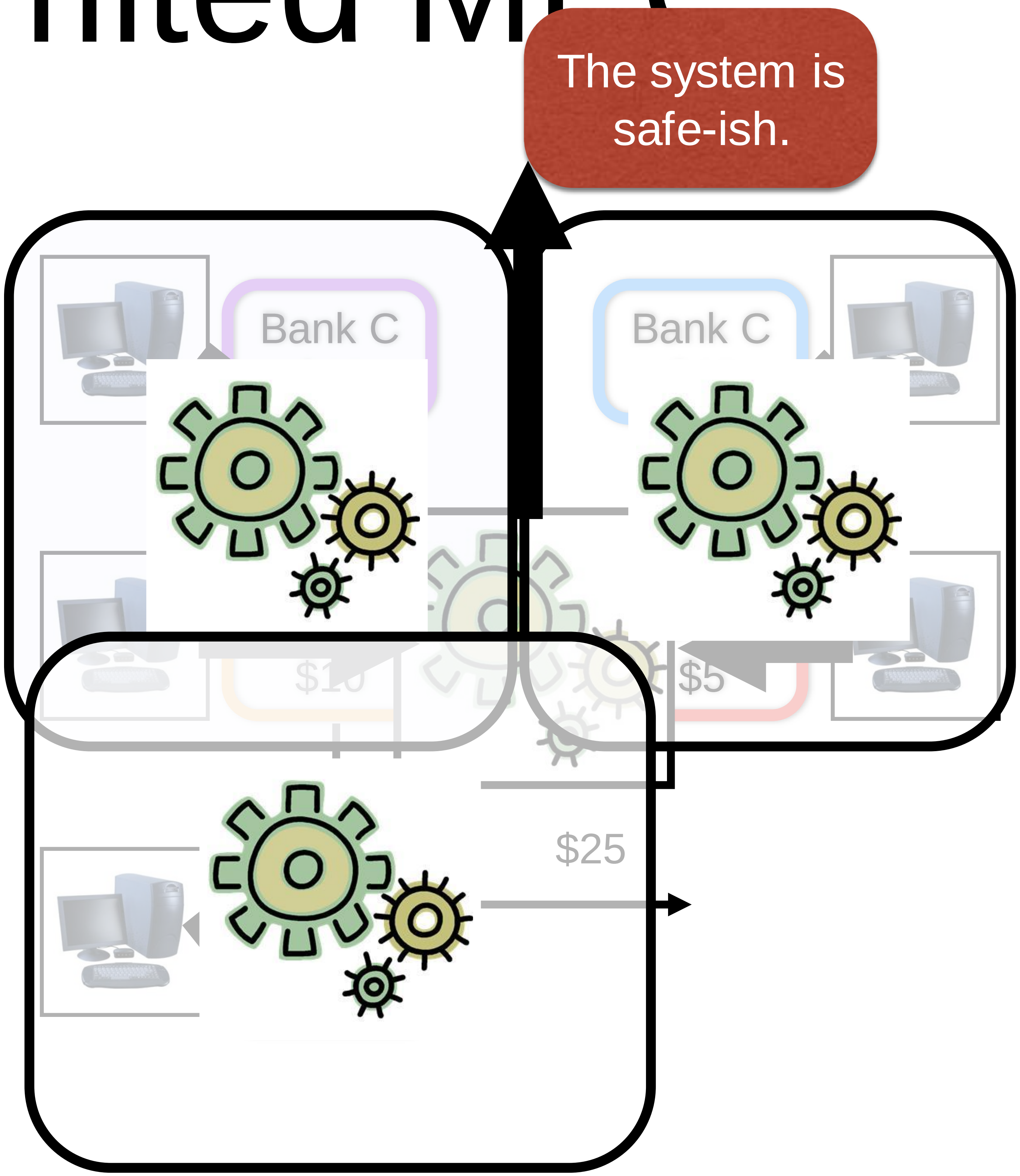
# Computing These Models

- Naively computing matrix multiplications in MPC won't work.
- Just as in PageRank…
- Iterative graph-based approaches are easier to execute…
- Especially when we take advantage of sparsity.



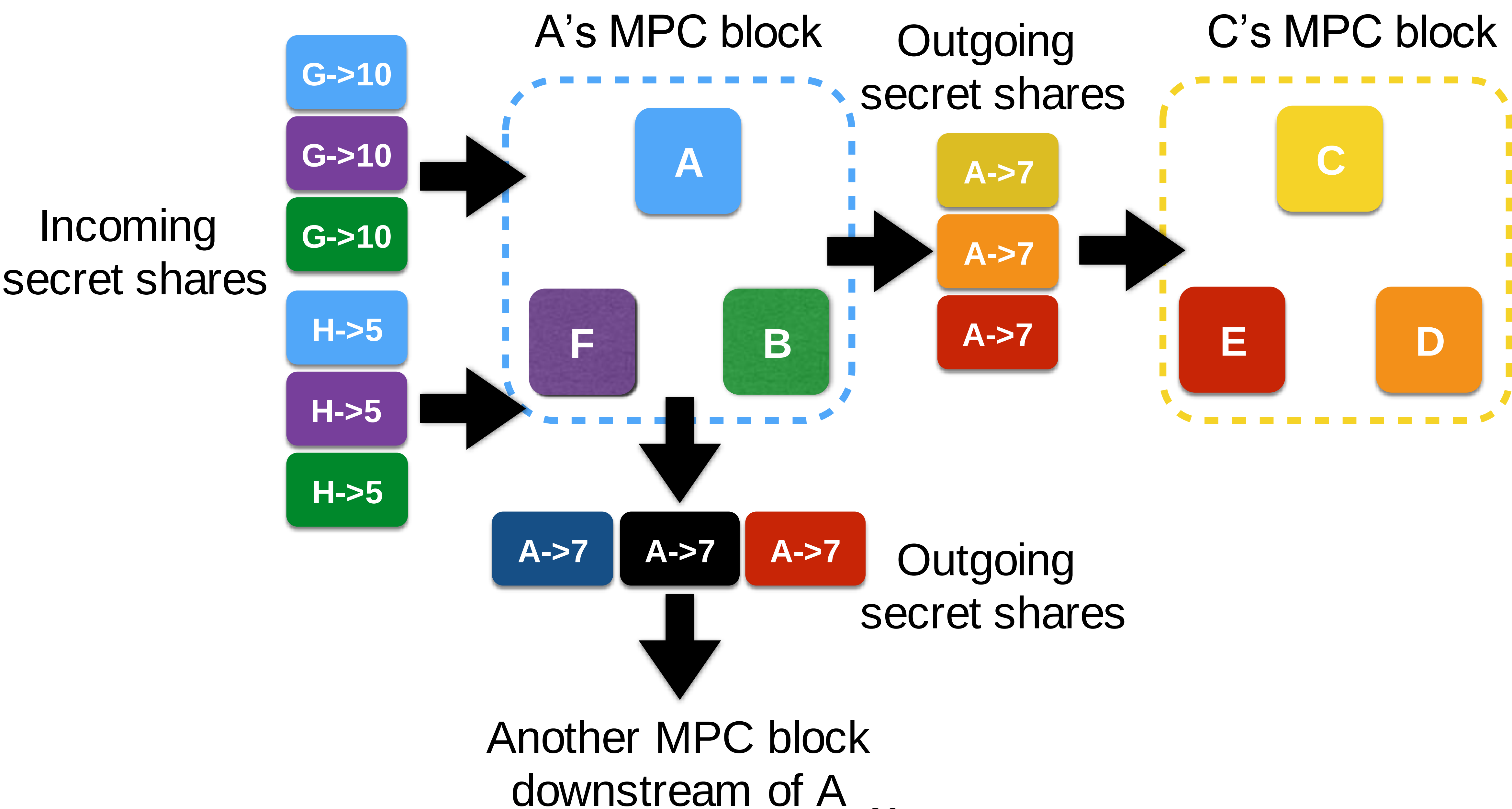
# Design: Limited MPC

- MPC with all parties is prohibitively expensive.
- Instead, we do multiple MPCs with sets of **k** parties.
- All intermediate state exists only as secret shares.
- The final aggregation adds differential privacy.



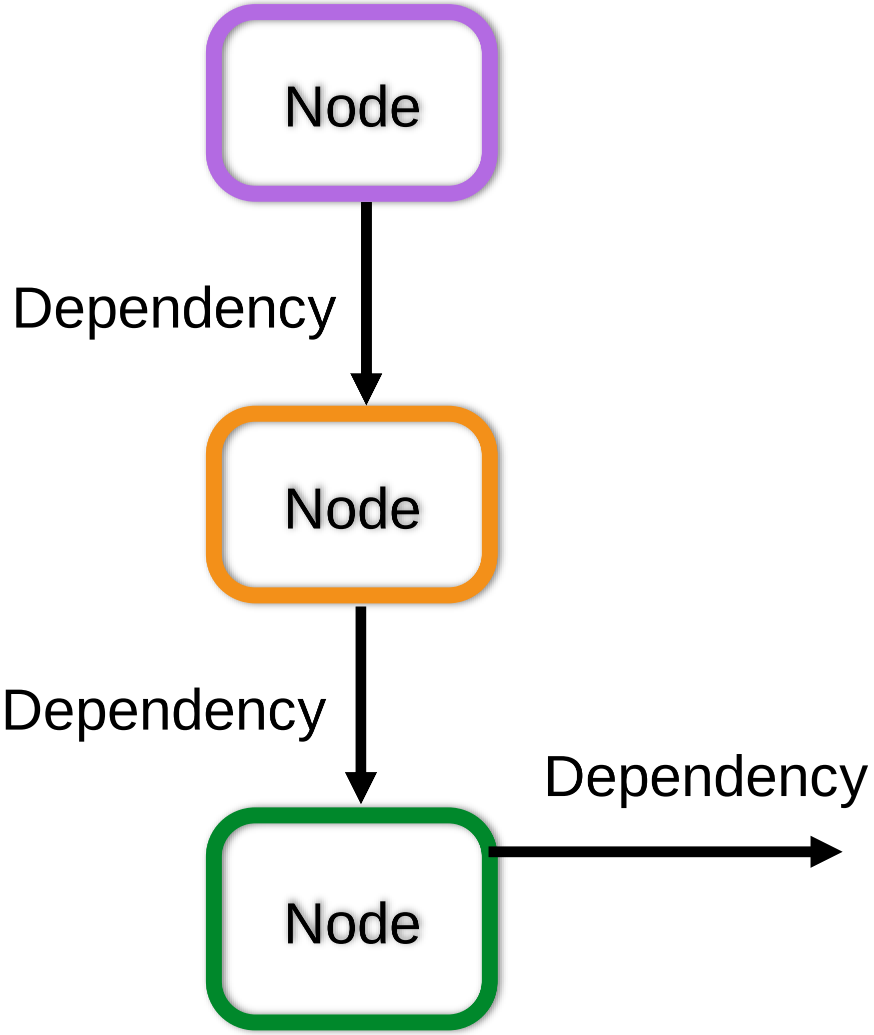
# Design: Secret Sharing

How do we keep the intermediate state private between MPC stages?



# Taking a step back…

- We have seen an important motivating scenario.
- We would have **Infrastructure** for privacy preserving graph-based computations.
- Banks can safely share their information with strong guarantees.
- Regulators can have a much better view into the system.



# Status and Ongoing Work

- We are building an implementation.
- Looking at a couple of economic models of contagion detection from the economics literature.
- Working on automatically certifying algorithms as differentially private.
- Other possible domains: BotNet detection?

# Summary

- Dependability is a broader challenge than technical systems.
- In this talk: dependability of the financial system.
- It has technical and economics aspects.
- Economists know **what** to compute, but not **how**.
- Key challenges: Privacy and Scalability.
- Our approach:
  exploit the graph structure, and use differential privacy