Trust and Accountability in Social Systems

Peter Druschel

MAX-PLANCK-GESELLSCHAFT

Max Planck Institute for Software Systems

Lecture overview

- Today’s computer systems augment a wide range of human activity, including cooperation among individuals, organizations, businesses
- This lecture deals with some of the challenges and opportunities that arise from this trend
- Specifically, we will talk about
  - mechanisms to provide accountability
  - how to leverage social connections to thwart undesired behavior

Lecture overview

1. Social Systems
   - What are they?
   - What is different about them?
2. Accountability for distributed systems
   - Why and what is accountability?
   - How can we implement it?
   - How well does it work?
3. Leveraging social relationships
   - Exploiting social networks for Sybil tolerance
   - Credit networks
   - Ostra: thwarting spam
   - Bazaar: limiting auction fraud
   - Genie: limiting social network crawling

Credits

Team members:

- Paarijaat Aditya
- Allen Clement
- Mainack Mondal
- Bimal Viswanath

Collaborators:

- Ioannis Avramopoulos, *DT Labs/TU Berlin*
- Krishna Gummadi, *MPI-SWS*
- Andreas Haeberlen, *UPenn*
- Petr Kuznetsov, *DT Labs/TU Berlin*
- Alan Mislove, *Northeastern*
- Ansley Post, *Google*
- Jennifer Rexford, *Princeton*
- Rodrigo Rodrigues, *MPI-SWS*

Social systems

Information system shared by autonomous users and organizations

- results depend on cooperation, good will
- vulnerable to misbehavior

Examples:

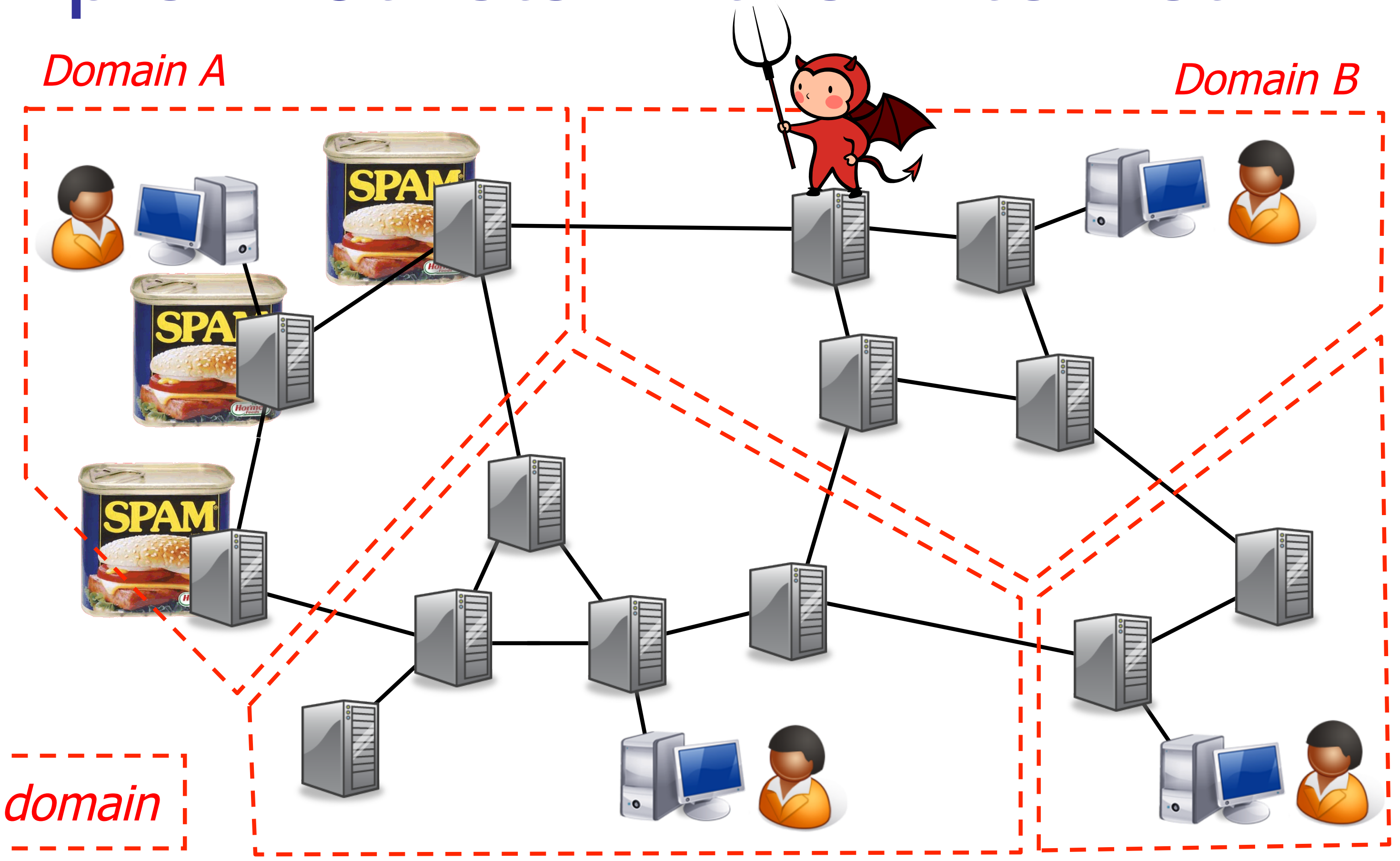
- Social production systems: Wikipedia, folksonomies, mechanical turk, open source
- Communication and sharing systems: Social networks, email, chat, (micro-)blogs
- Federated systems: Internet, WWW, …

Social systems: What’s different?

- *Conventional systems:* operating environment can be modeled fully
- *Adaptive systems engineering:* operating environment cannot be fully modeled a priori; system is able to learn from experience with the environment
- *Social systems:* environment (users) adapts to the system and may even play an adversarial role
  - must consider incentives for participants
  - must anticipate deviant behavior

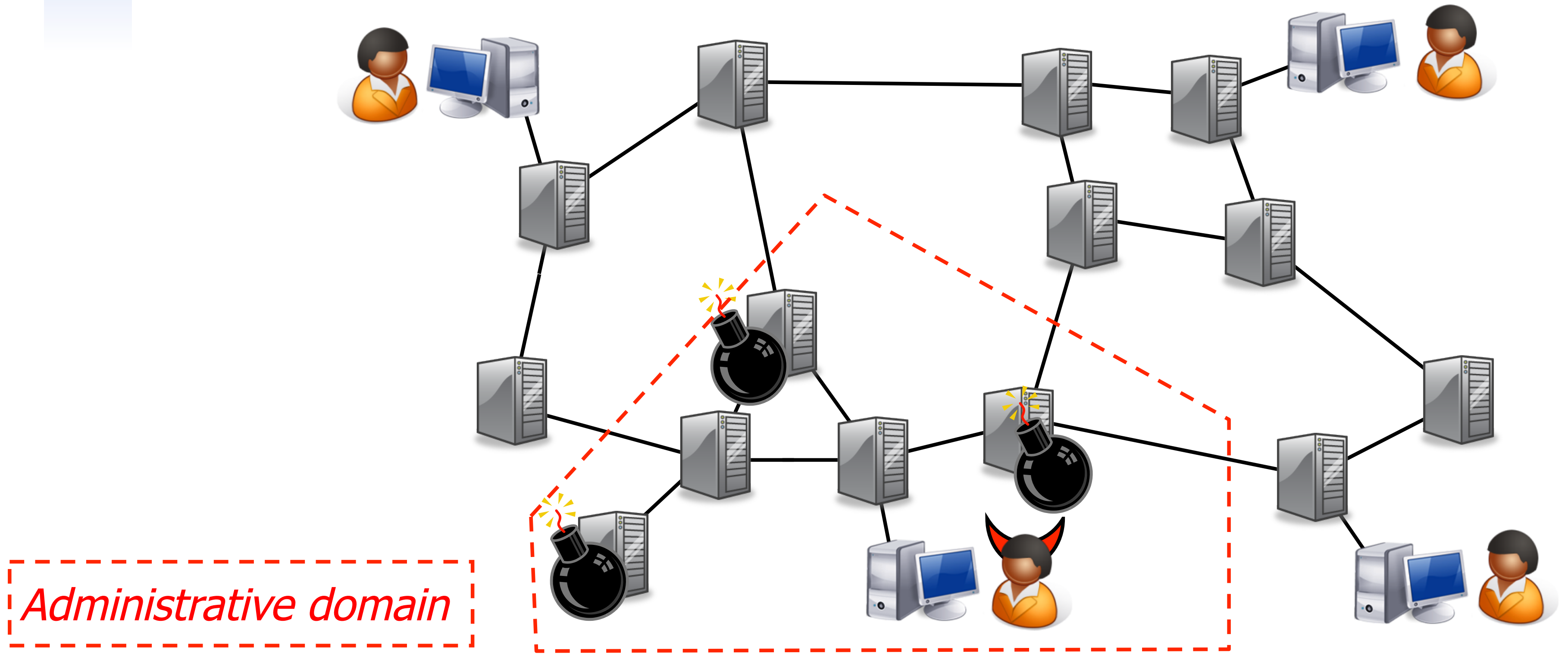
Outline

Example: Botnets in the Internet



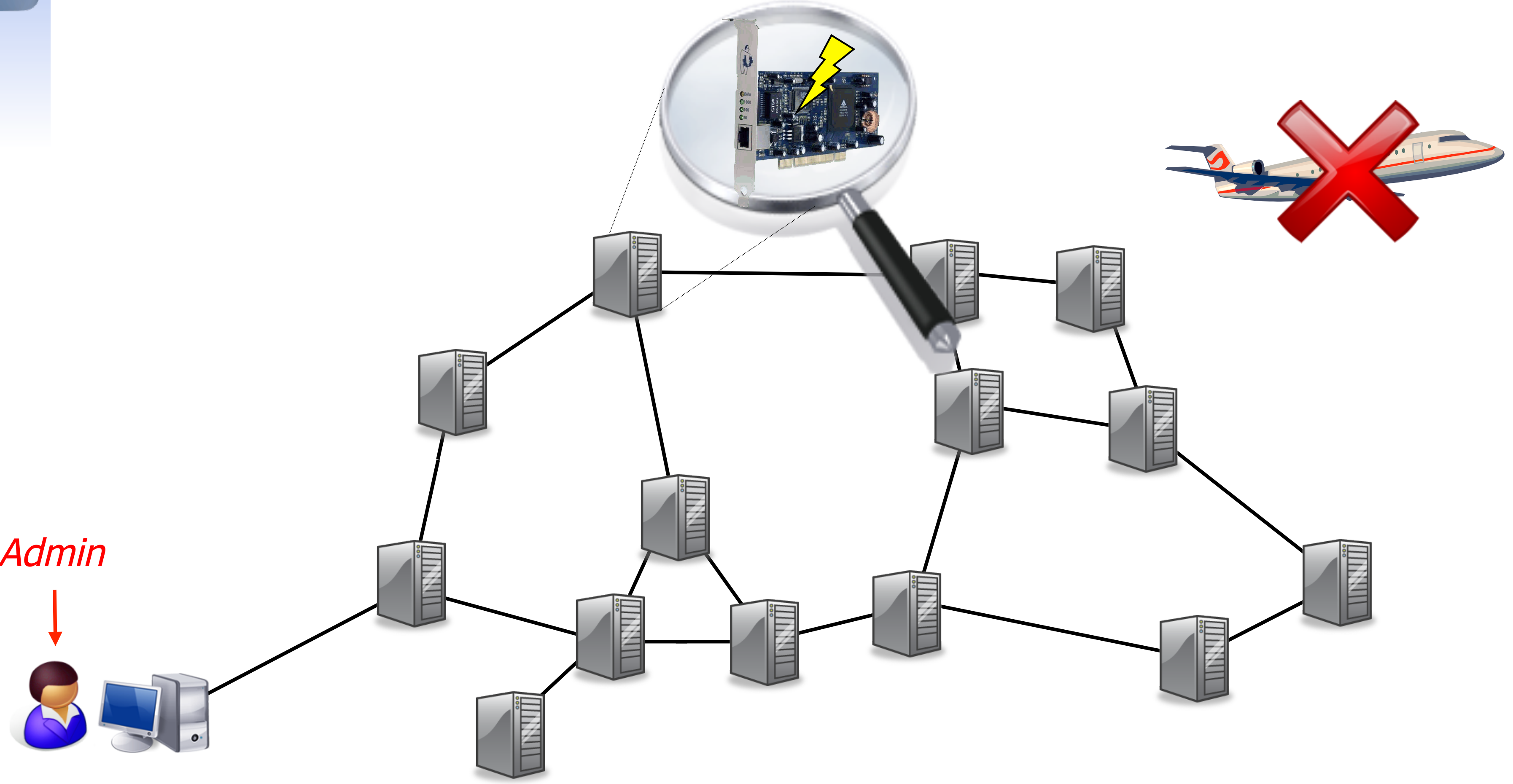
- Compromised computer targets different domain
- Admin *A* must localize fault, then convince admin *B* that her machine is faulty

Example: Insider attack



- Mar 2002: UBS PaineWebber admin disrupts trade for days to weeks
- Difficult to detect, defuse logical bombs

Example: LAX airport outage



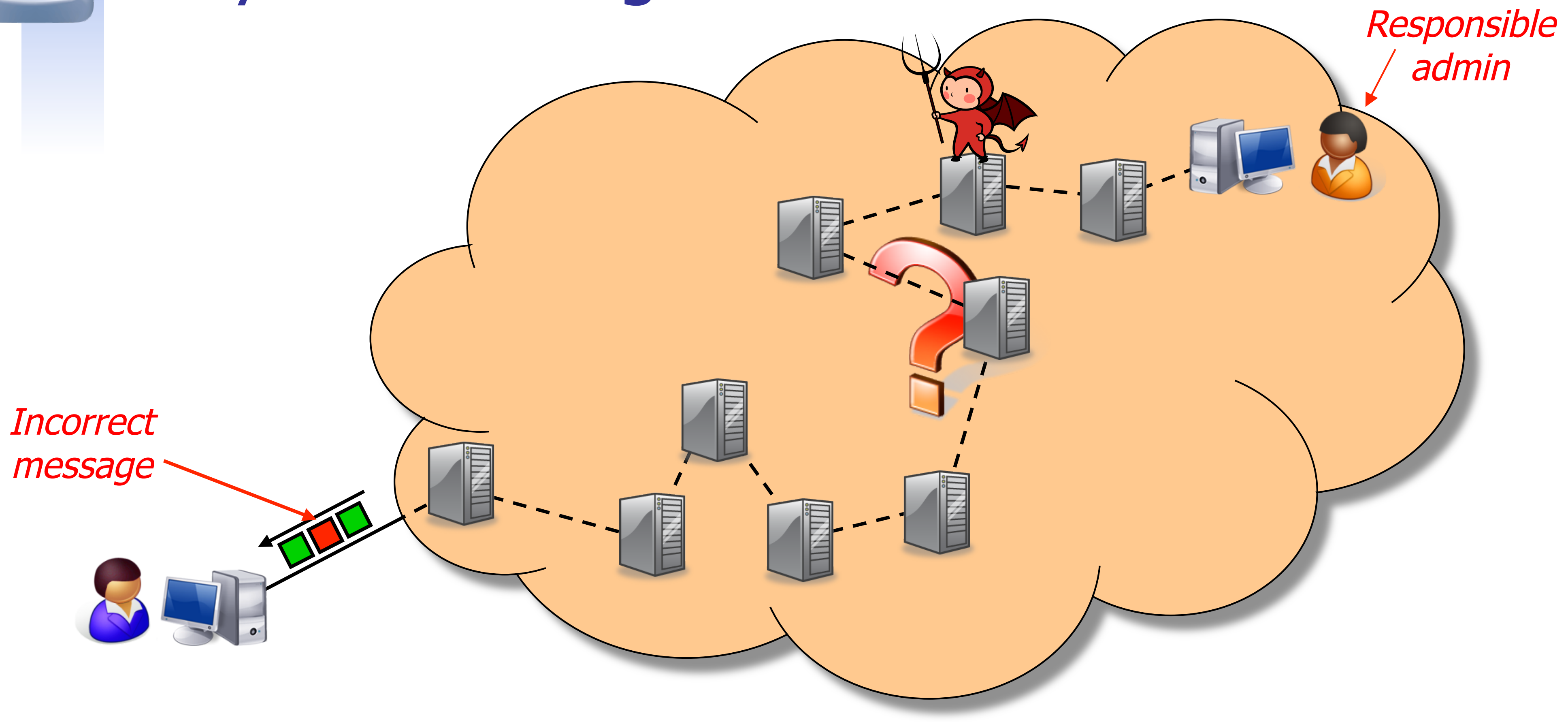
- Aug 2007: 17,000 passengers stranded at LAX
- Cause: intermittent fault of a network card

Focus: Byzantine faults

- Not all faults cause a node to stop
  - The faulty node continues to operate, but its behavior deviates from that of a correct node

- Examples:
  - Hardware malfunction
  - Misconfiguration
  - Software error
  - External security attack
  - Intentional software modification

Why is detecting faults difficult?



- How to detect faults?
- How to identify the faulty node?
- How to convince others that a node is (not) faulty?

Learning from the 'offline' world

- Relies on accountability
- Example: Banks

Requirement	Solution
Commitment	Signed receipts
Tamper-evident record	Double-entry bookkeeping
Inspections	Audits

- Record can be used to (manually) detect, identify and convince
- Is accountability useful in distributed systems?
- Is it practical?

What does accountability mean?

Accountability := *tamper-evident* record + *automated, reliable* fault detection

Is accountability alone useful?

No, if faults are severe and irrecoverable

- need byzantine fault *tolerance* (different lecture)

Yes, for

- systems that provide „best-effort“ service
- systems that mask severe/irrecoverable faults
- systems that assume crash failures

Accountability

- reliably detects and localizes faults
- provides incentives to avoid faults
- builds trust, reputation

Butler Lampson on accountability

"Don’t forget that in the real world, security depends more on police than on locks, so detecting attacks, recovering from them, and punishing the bad guys are more important than prevention."

-- Butler Lampson, "Computer Security in the Real World", ACSAC 2000

Outline

1. Social systems
2. Accountability for distributed systems
   - Why accountability?
   - What is accountability?
   - How can we implement it?
   - How well does it work?
   - Challenges and extensions
3. Leveraging social relationships

Ideal accountability

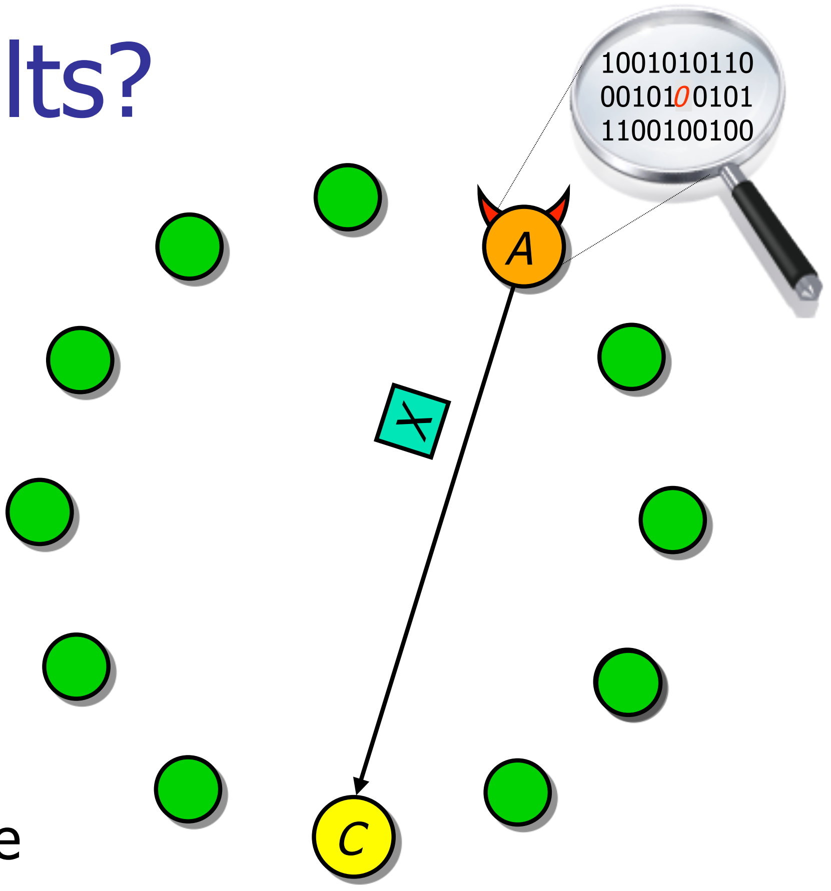
- Fault := Node deviates from expected behavior
- Our goal is to *automatically*
    - detect faults
    - identify the faulty nodes
    - convince others that a node is (or is not) faulty

- Can we build a system that provides the following guarantee?

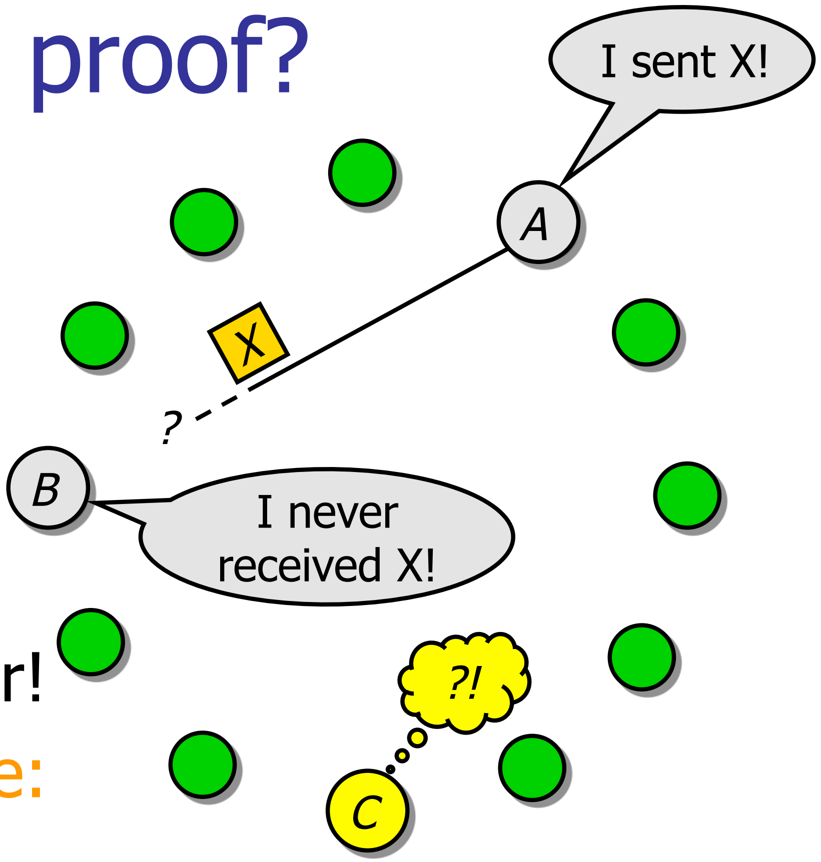
Whenever a node is faulty in any way, the system generates a proof of misbehavior against that node

Can we detect all faults?

- Problem: Faults that affect only a node's internal state
  - Would require online trusted probes at each node
- Focus on observable faults:
  - Faults that affect a correct node
- Can detect observable faults without requiring trusted components



Can we always get a proof?



- Problem: He-said-she-said
- Three possible causes:
  - A never sent X
  - B refuses to acknowledge X
  - X was lost by the network
- Cannot get proof of misbehavior!
- Generalize to verifiable evidence:
  - a proof of misbehavior, or
  - a challenge that a faulty node cannot answer
- What if the challenged node does not respond?
  - Does not prove a fault, but node is suspected until it responds

Practical accountability

- We propose the following requirement for an accountable distributed system:

  Whenever a fault is observed by a correct node, the system eventually generates verifiable evidence against a faulty node

- This is useful
  - Any (!) fault that affects a correct node is eventually detected and linked to a faulty node

- It can be implemented in practice (as we will see)

Outline

1. Social systems
2. Accountability for distributed systems
   - Why accountability?
   - What is accountability?
   - How can we implement it?
   - How well does it work?
   - Challenges and extensions
3. Leveraging social relationships

An implementation: PeerReview

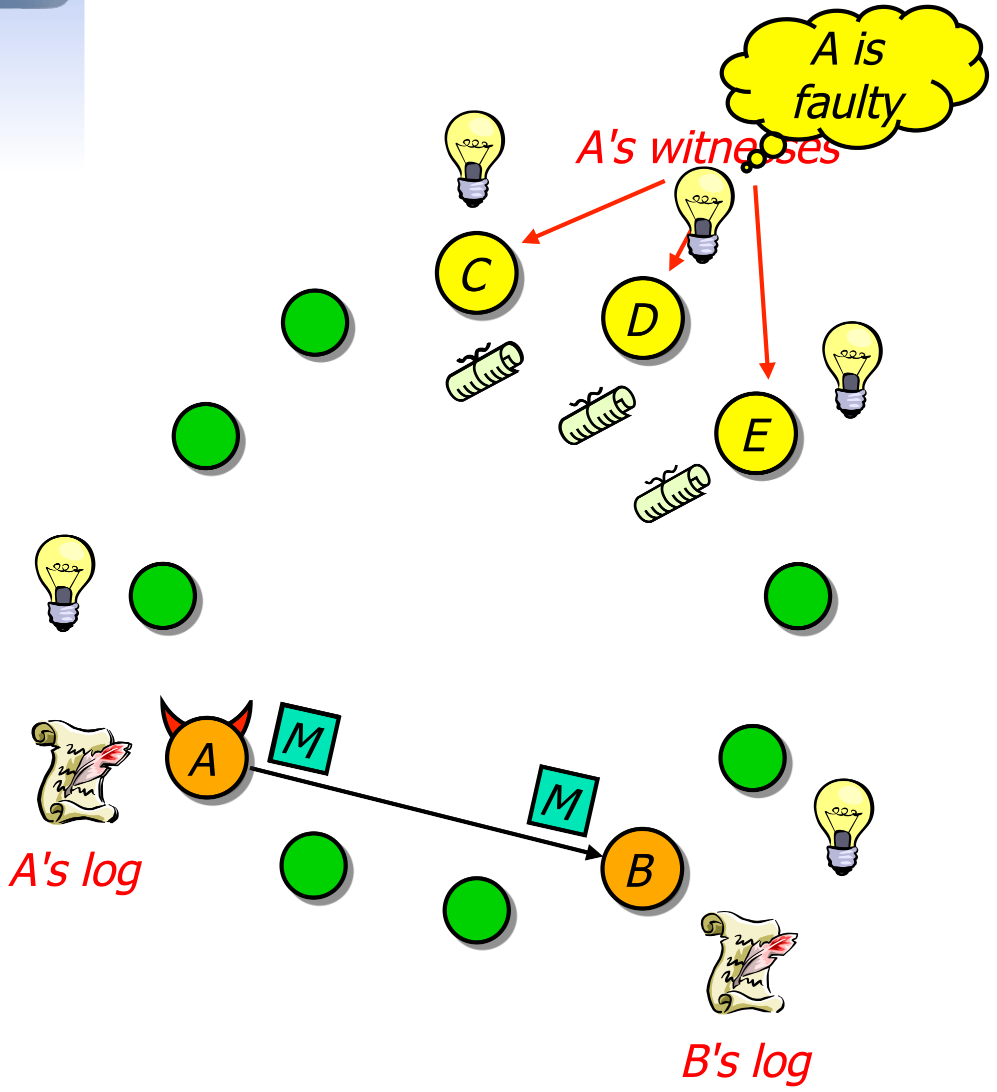
Adds accountability to a given system

- Implemented as a library
- Provides tamper-evident record
- Detects faults via state-machine replay

Assumptions:

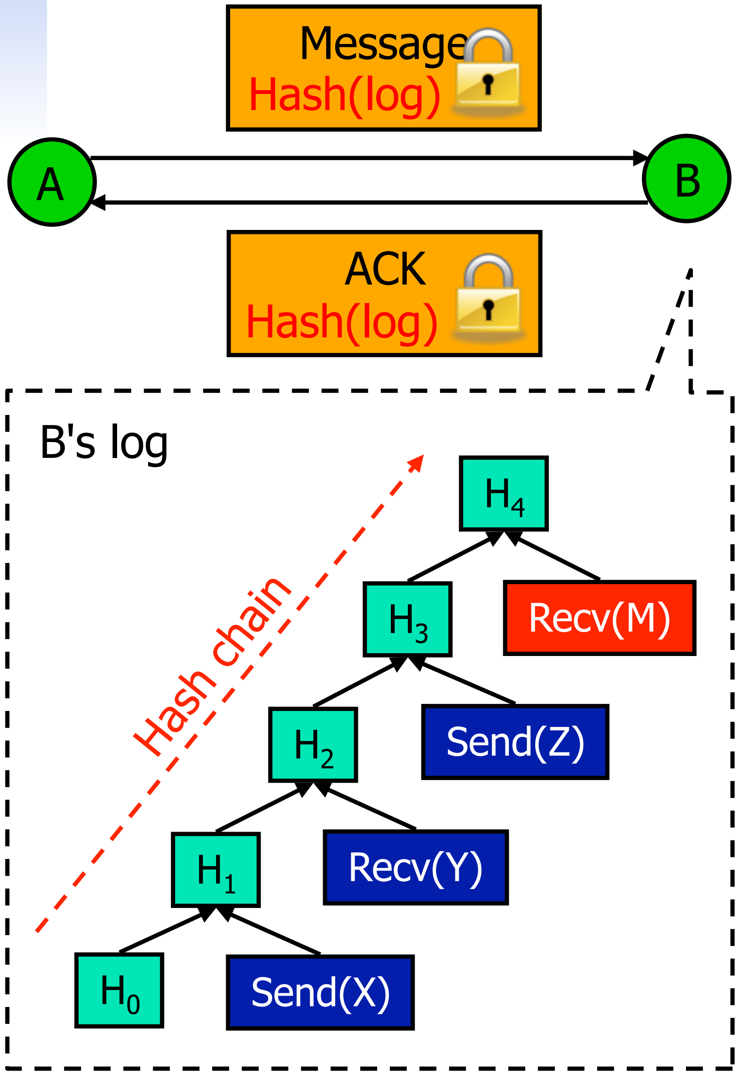
1. Nodes can be modeled as deterministic state machines
2. Nodes have reference implementations of the state machines
3. Correct nodes can eventually communicate
4. Nodes can sign messages

PeerReview from 10,000 feet



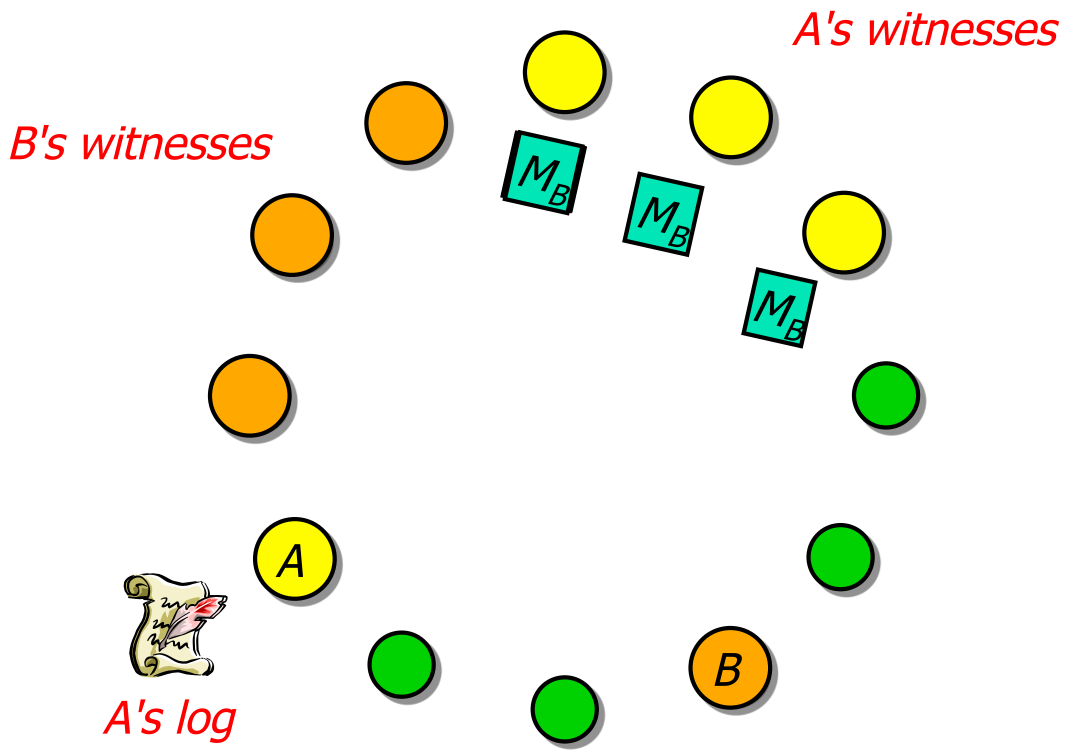
- All nodes keep logs of their inputs & outputs
  - Including all messages
- Each node has a set of witnesses, which audit the node periodically
- If the witnesses detect misbehavior, they
  - generate evidence
  - make the evidence available to other nodes
- Other nodes check evidence, report fault

PeerReview detects tampering



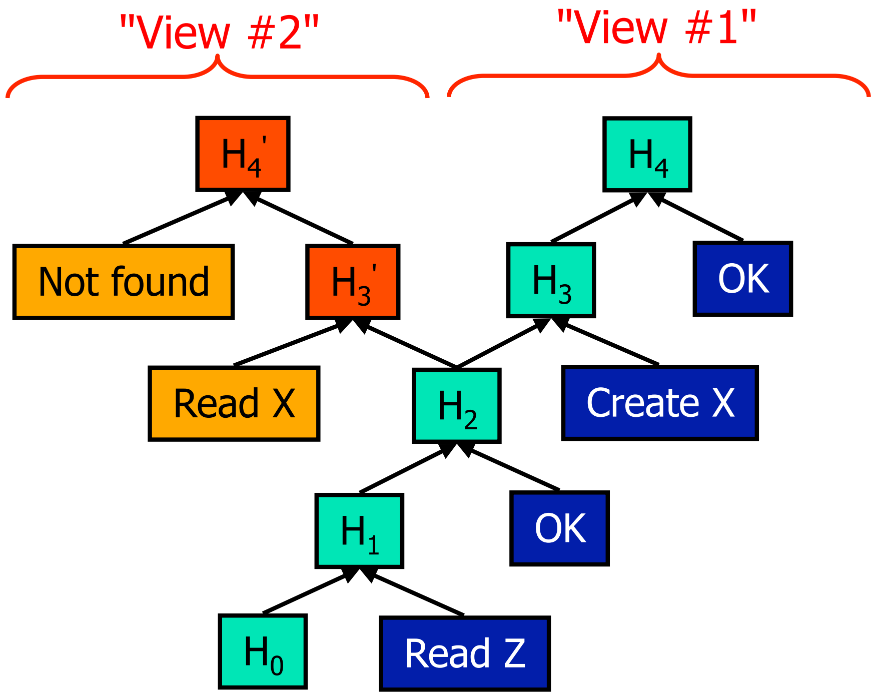
- What if a node modifies its log entries?
- Log entries form a hash chain
  - Inspired by secure histories [Maniatis02]
- Hash is included with every message authenticator
  - ⇒ Node commits to its current state
  - ⇒ Changes are evident

PeerReview detects omission



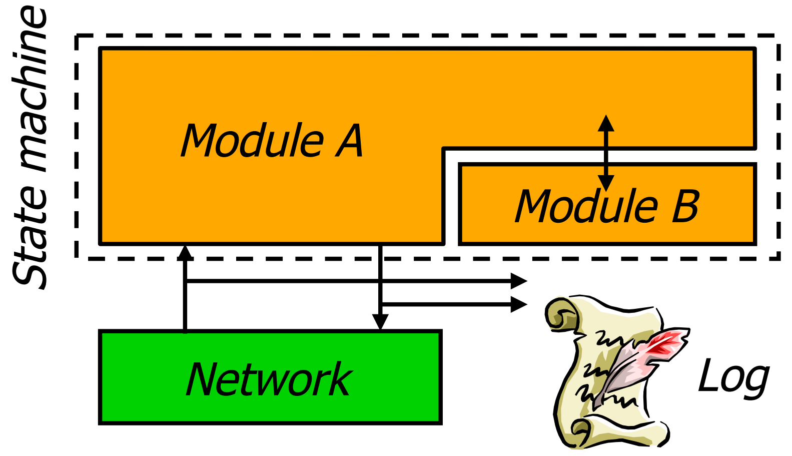
- What if a node omits log entries?
- While inspecting A’s log, A’s witnesses send msg authenticators signed by B to B’s witnesses
- Thus, witnesses learn about all messages their node has ever sent or acknowledged
- Omission of a message from the log is a fault

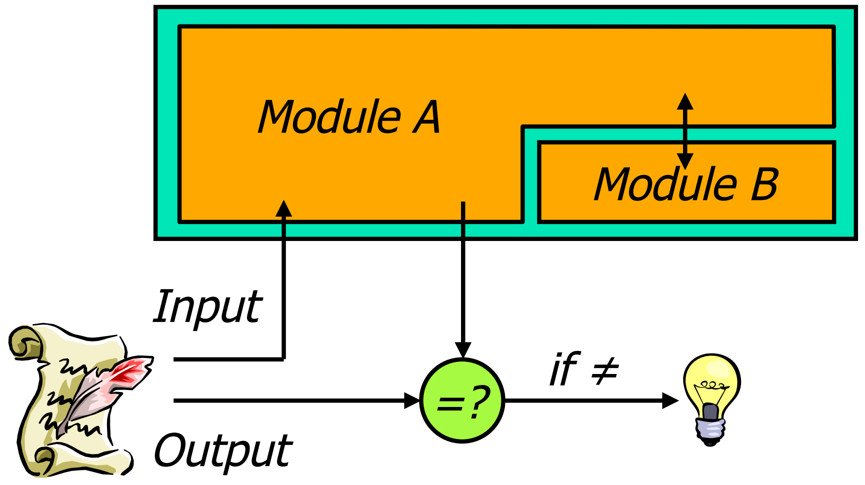
PeerReview detects inconsistencies



- What if a node
  - keeps multiple logs?
  - forks its log?
- Witnesses check whether all msg authenticators form a single hash chain
- Two authenticators not connected by a log segment indicate a fault

PeerReview detects faults





- How to recognize faults?
- Assumption:
  - Nodes can be modeled as deterministic state machines
- To audit a node, witness
  - Fetches signed log
  - Replays inputs to a trusted copy of the state machine
  - Checks outputs against the log

PeerReview guarantees

1) **Observable faults will be detected**

If node commits a fault + has a correct witness, then witness obtains

- *a proof of misbehavior (PoM), or*
- *a challenge that the faulty node cannot answer*

2) **Good nodes cannot be accused**

If node is correct

- *there can never be a PoM, and*
- *it can answer any challenge*

- Formal analysis in [*TR MPI-SWS-2007-003*] see also [*Haeberlen&Kuznetsov, OPODIS'09*]

PeerReview is widely applicable

- App #1: NFS server in the Linux kernel
  - Many small, latency-sensitive requests
    - Tampering with files
    - Lost updates
    - *Metadata corruption*
    - *Incorrect access control*
- App #2: Overlay multicast
  - Transfers large volume of data
    - Freeloading
    - Tampering with content
- App #3: P2P email
  - Complex, large, decentralized
    - Denial of service
    - Attacks on DHT routing
    - *Censorship*
- More information in [*Haeberlen et al., SOSP'07*]

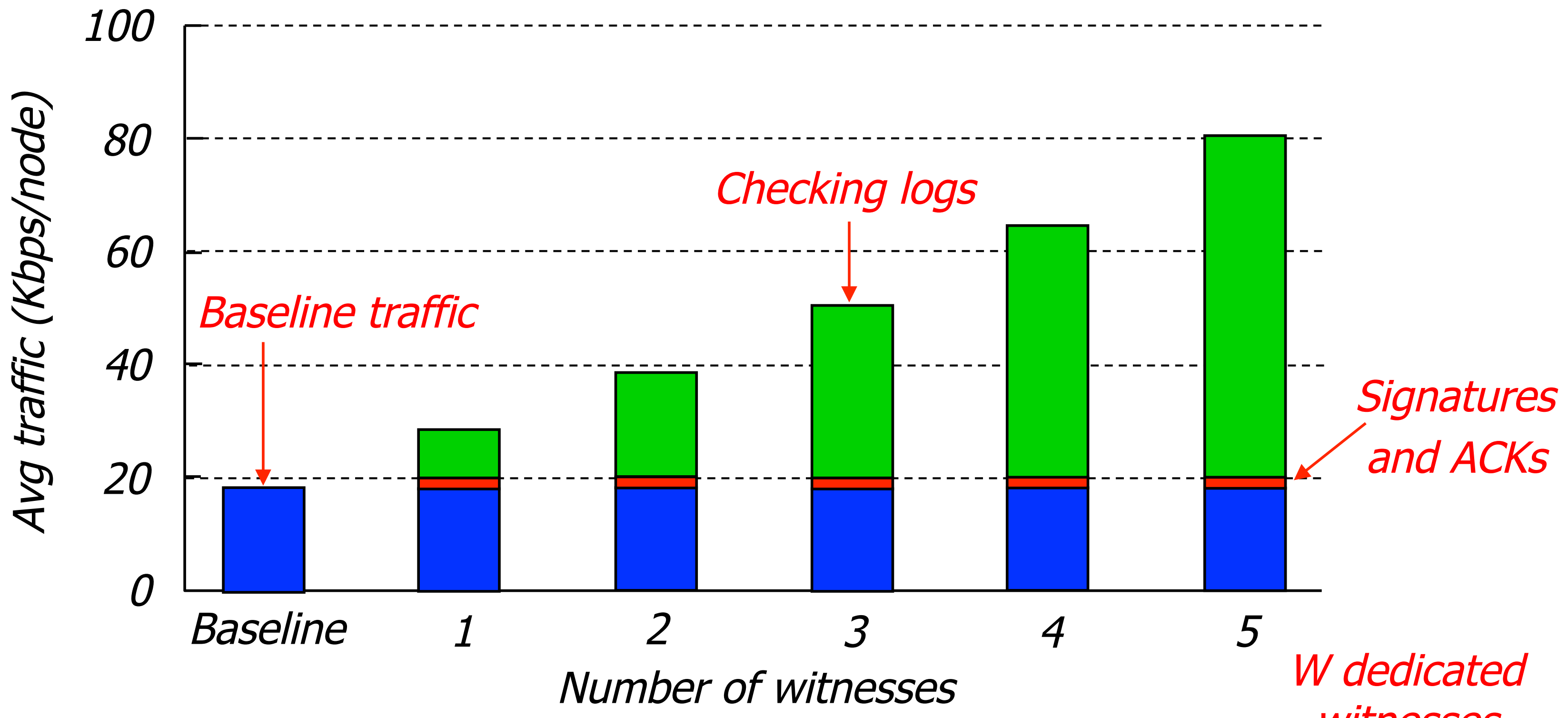
Outline

1. Social systems
2. Accountability for distributed systems
   - Why accountability?
   - What is accountability?
   - How can we implement it?
   - How well does it work?
   - Challenges and extensions
3. Leveraging social relationships

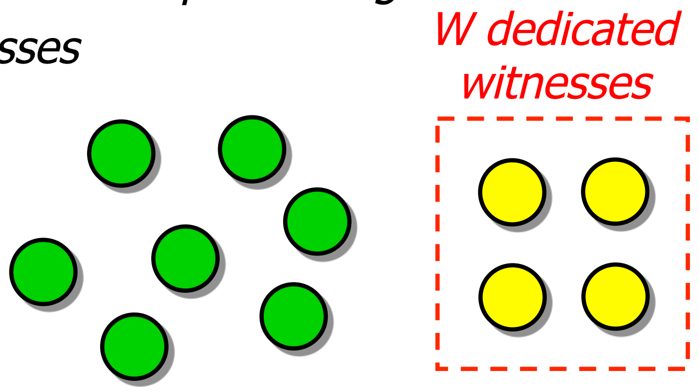
How much does PeerReview cost?

- Log storage
  - 10 – 100 GByte per month, depending on application
- Message signatures
  - Message latency (e.g. 1.5ms RTT with RSA-1024)
  - CPU overhead (embarrassingly parallel)
- Log/authenticator transfer, replay overhead
  - Depends on # witnesses
  - Can be deferred to exploit bursty/diurnal load patterns

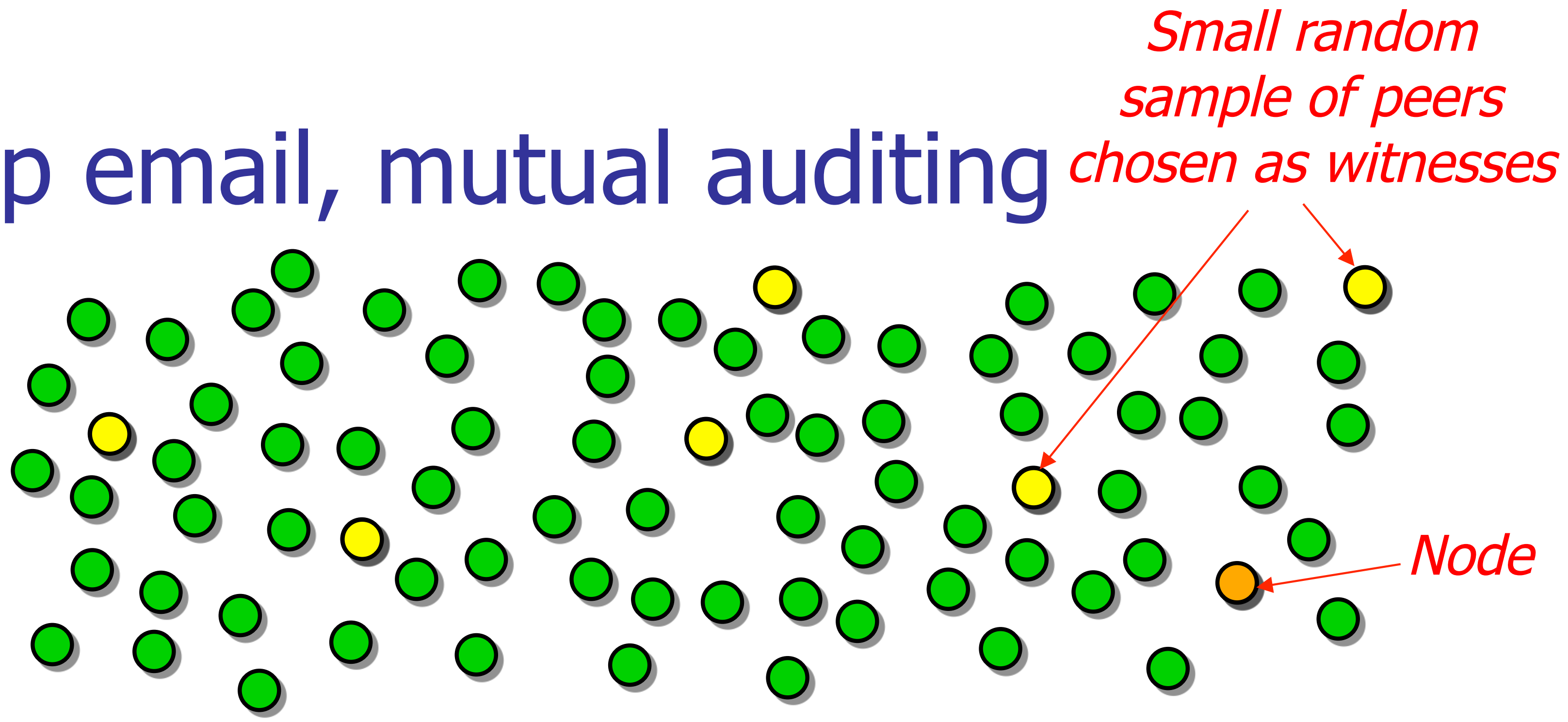
P2p email, dedicated witnesses



- Dominant cost depends on number of witnesses W
  - O(W^2) component

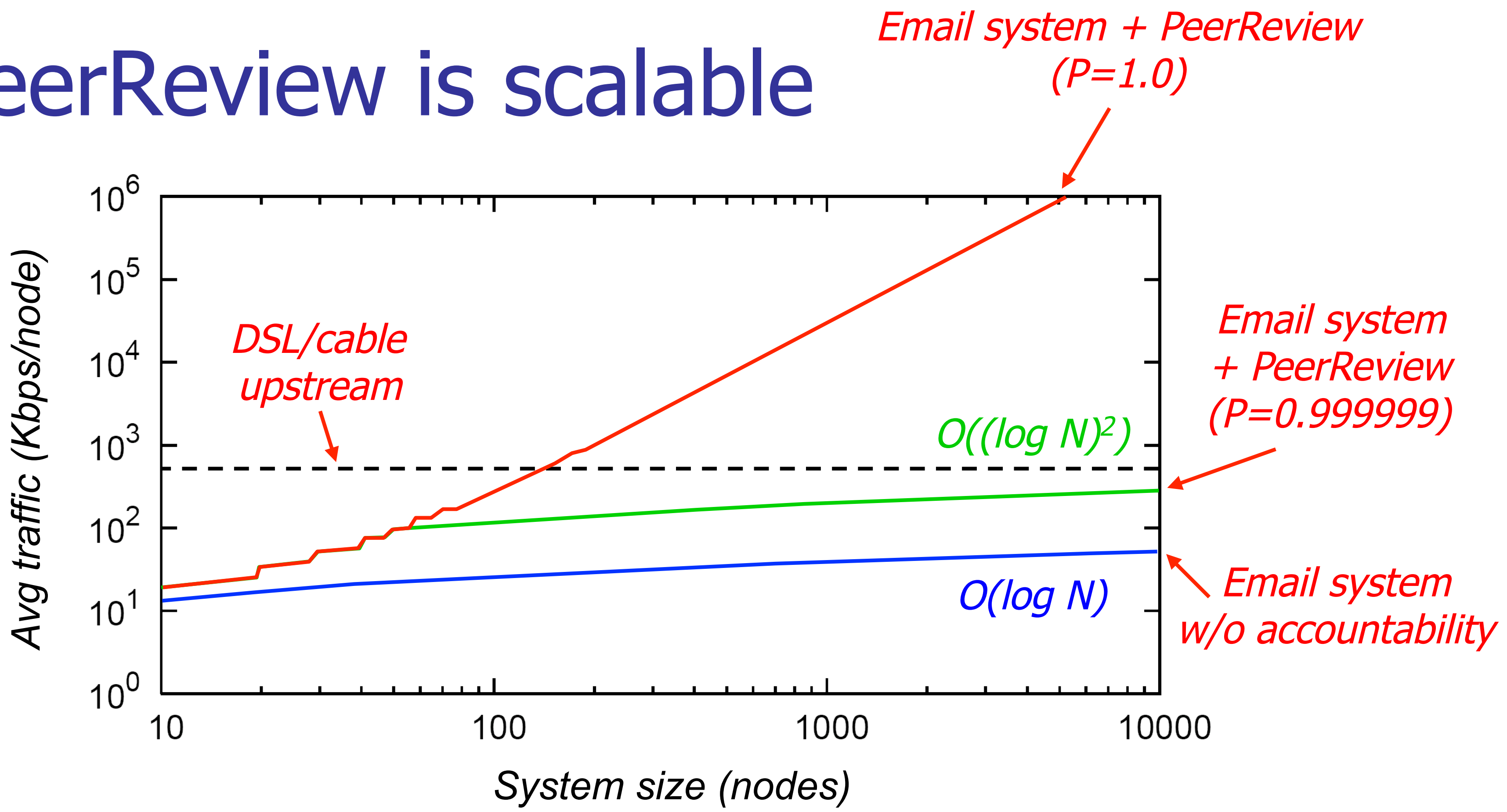


P2p email, mutual auditing



- Small probability of error is inevitable
  - Example: Replication
- Can use this to optimize PeerReview
  - Accept that an instance of a fault is found only with high probability
  - Asymptotic complexity: $O(N^2) \rightarrow O(\log N)$

PeerReview is scalable



- Assumption: up to 10% of nodes can be faulty
- Probabilistic guarantees provide scalability
  - Example: email system scales to over 10,000 nodes with P=0.999999

PeerReview summary

- Accountability is a new approach to handling faults in distributed systems
  - detects faults
  - identifies the faulty nodes
  - produces evidence
- PeerReview: A library and system that provides accountability
  - Offers provable guarantees and is widely applicable

Details in [*Haeberlen et al., SOSP '07*]

Outline

1. Social systems
2. Accountability for distributed systems
   - Why accountability?
   - What is accountability?
   - How can we implement it?
   - How well does it work?
   - Challenges and extensions
3. Leveraging social relationships

Challenges

- Tension between accountability and privacy
  - PeerReview (PR) requires disclosure to witnesses
  - Zero-knowledge proofs instead?
  - Accountable randomness [*Backes et. al., NDSS'09*]

- Fault detection
  - PR uses state-machine replay for fault detection
  - Can't detect deterministic software bugs
  - Different implementations of underspecified protocols may diverge
  - Protocol specification or abstract model instead?

Challenges (cont‵d)

- Message signatures
  - PR assumes a public-key infrastructure
  - Web-of-trust (physical network, social network) instead?
- Partial deployment
  - Accountability zones, gateways?
- PR requires source code modifications
  - To enable deterministic replay
  - Accountable virtual machines instead?

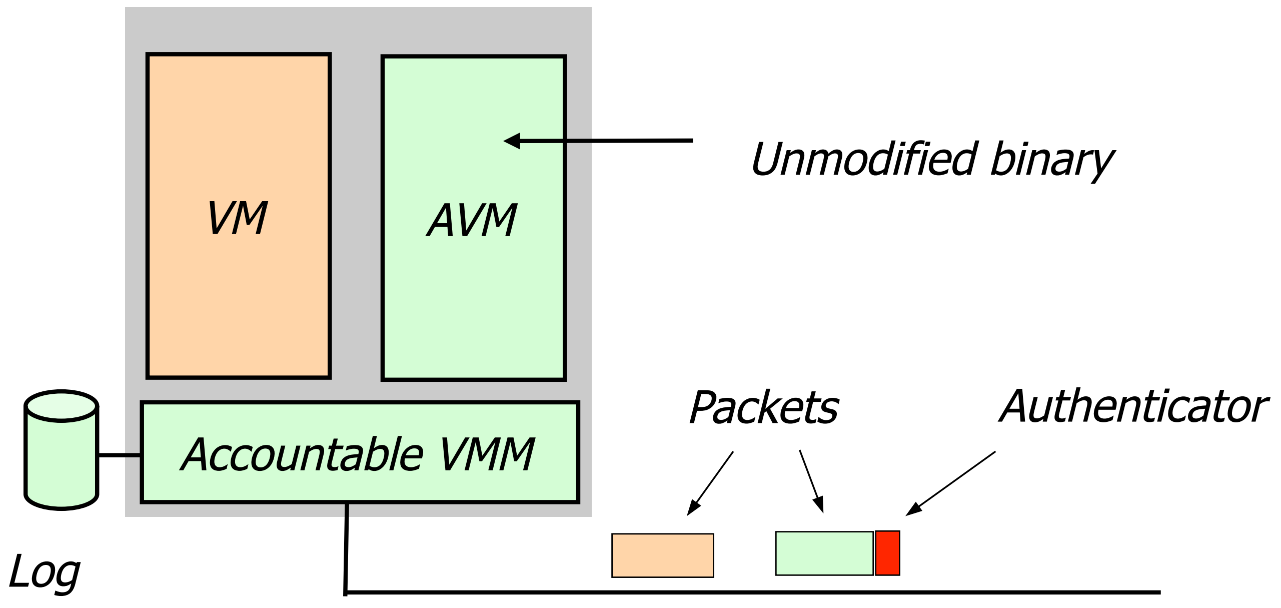
NetReview

- Accountability applied to inter-domain routing
- Fault detection based on a specification of the routing protocol and policy
- Web-of-trust-based certificates
- Auditing limited to peering partners
- Partial deployment: accountability zones

Details in [*Haeberlen* et. al., *NSDI'09*]

Accountable virtual machines (AVM)

- Make unmodified binary VMs accountable
- VMM provides deterministic logging/replay



- Details in [*Haeberlen et al., OSDI 2010*]

Related Work

- Accountability [*Lampson '00, Yumerefendi&Chase '05, Yemerefendi et al. '07, Argyraki et al. '07, Michalakis et al. '07*]
- Practical byzantine fault tolerance [*Castro&Liskov '00, Ramasamy '07*]
- General fault detection [*Kihlstrom et al. '07, Doudou et al. '99, Malkhi&Reiter '97*]
- Intrusion detection, reputation systems [*Denning '87, Ko et al. '94, Kamvar et al. '03*]
- Trusted computing [*Garfinkel et al. '02*]
- Fault-specific defenses [*Cox&Noble '03, Waldman&Mazieres '03*]
- Tamper-evident logs [*Schneier&Kelsey '98, Maniatis&Baker '02*]

Conclusion

- Byzantine faults in distributed systems are real

- Accountability is a new approach to handling faults
  - detects observable faults
  - identifies the faulty node
  - produces verifiable evidence

- Practical implementations exist

Outline

1. Social systems
2. Accountability for distributed systems
3. Leveraging social relationships
   - Exploiting social networks for Sybil tolerance
   - Credit networks
     - Ostra: thwarting spam
     - Bazaar: limiting auction fraud
     - Genie: limiting social network crawling

Sybil attacks

- Fundamental problem in systems with weak user ids
  - Social networks, eBay, gmail, p2p, etc.
- An individual who controls several identities
  - Can shed bad reputations (whitewashing)
  - Can manipulate reputation/history via fake transactions
  - Can manipulate voting
  - Can circumvent per-user limits
  - Can undermine fault model
- Examples:
  - Content vote tampering on YouTube, Digg
  - eBay fraud

Sybil defense approaches

Link user accounts to a more-or-less hard to obtain resource

1. Certification from trusted authorities
   - E.g., passport, social security number, credit card
   - May remove ability to use pseudonyms, official/private ids
2. Require work or money
   - Vulnerable to “rich” attackers
   - E.g., deep pocket attackers, botnets, rented cloud computing resources
3. Leverage social links

All may increase barriers to sign-up!

Two approaches to social network based Sybil defense

- Sybil detection: Identi
- Sybil tolerance: Limit the

Reputation systems
[Levien, USENIX Sec' 98],
[Cheng, P2PECON' 05],
[Ziegler, Inf. Sys.Front.' 05]

Ostra *[Mislove, NSDI' 08]*

SumUp [*Tran, NSDI'09*]

Bazaar [Post, NSDI' 11]

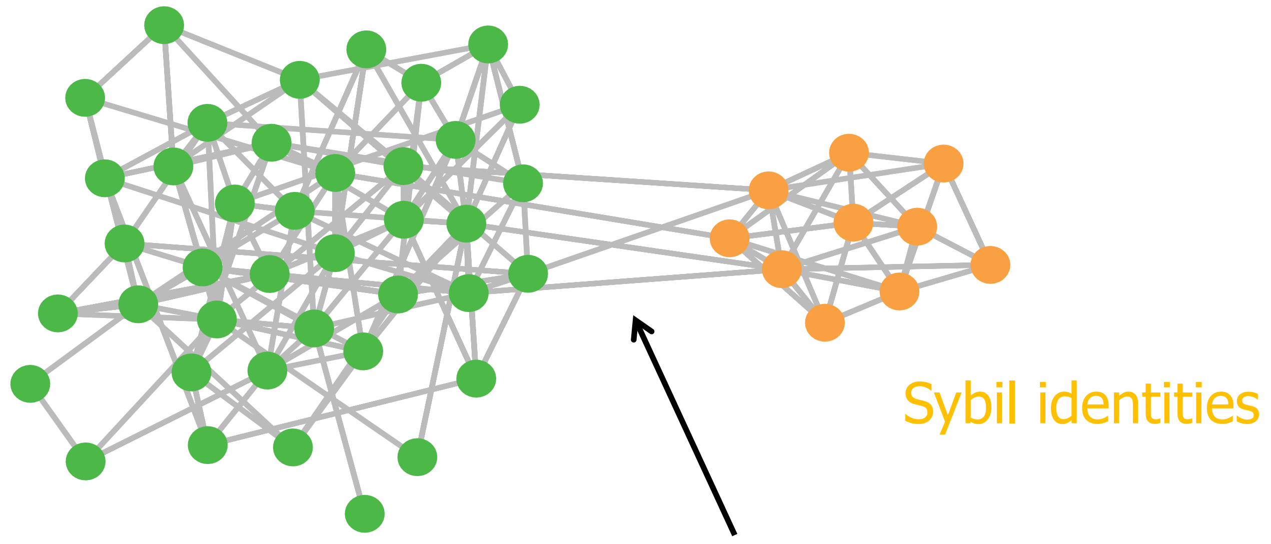
Genie [Mondal, SIGCOMM' 11 poster]

GateKeeper [INFOCOM' 11]

Links in social networks

Assumption: Links take some effort to form and maintain

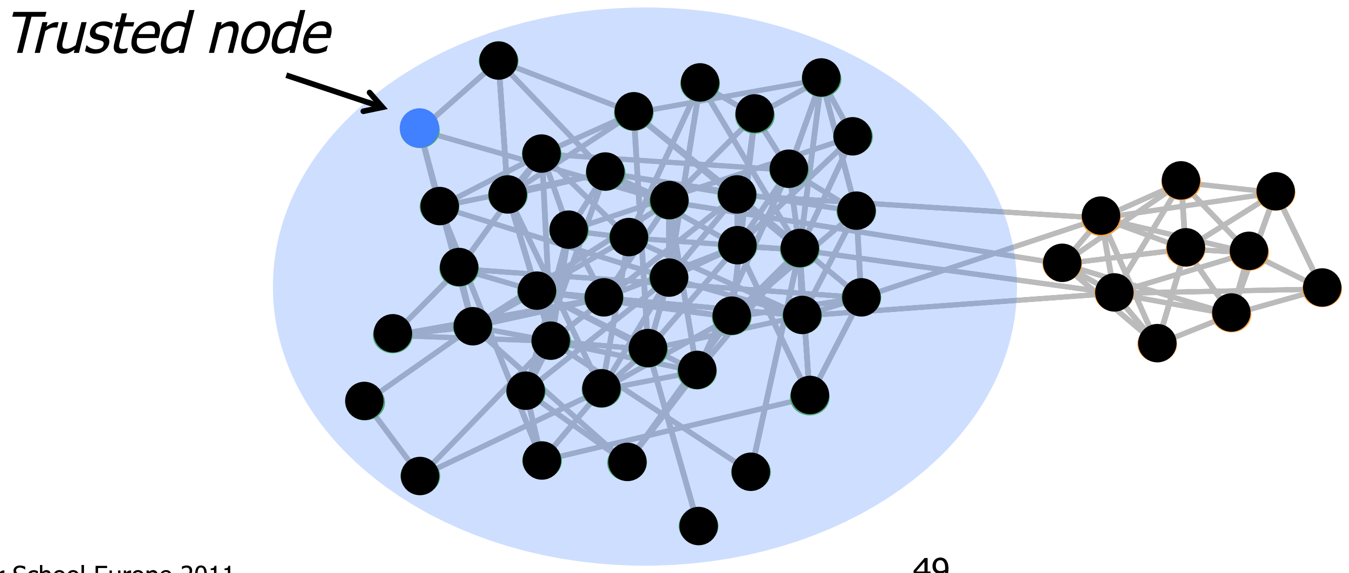
E.g.: Good users only accept links from users they recognize



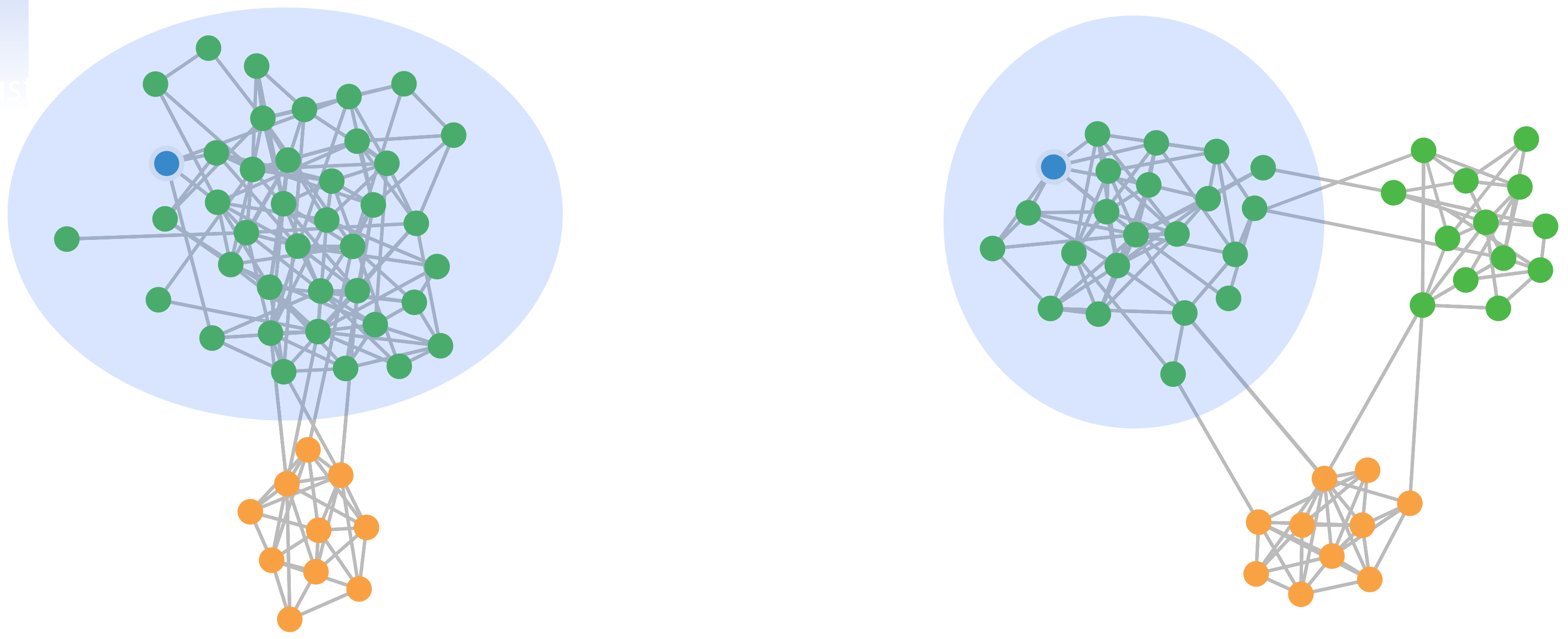
Attacker is limited by his ability to form social links to real users

Understanding Sybil detection

- All schemes work in a similar manner [*Viswanath et al., SIGCOMM10*]
- Effectively, they detect *communities*
- Nodes within a trusted node's community presumed good
- Other nodes presumed to be Sybils



Limitations of Sybil detection



- Detectors assume social networks are fast mixing
- Lots of evidence that many social networks have small fringe communities [*Leskovec 2008*], [*Dell'Amico 2009*]
- Sybil clouds and small communities may be indistinguishable using the graph structure alone

Sybil tolerance

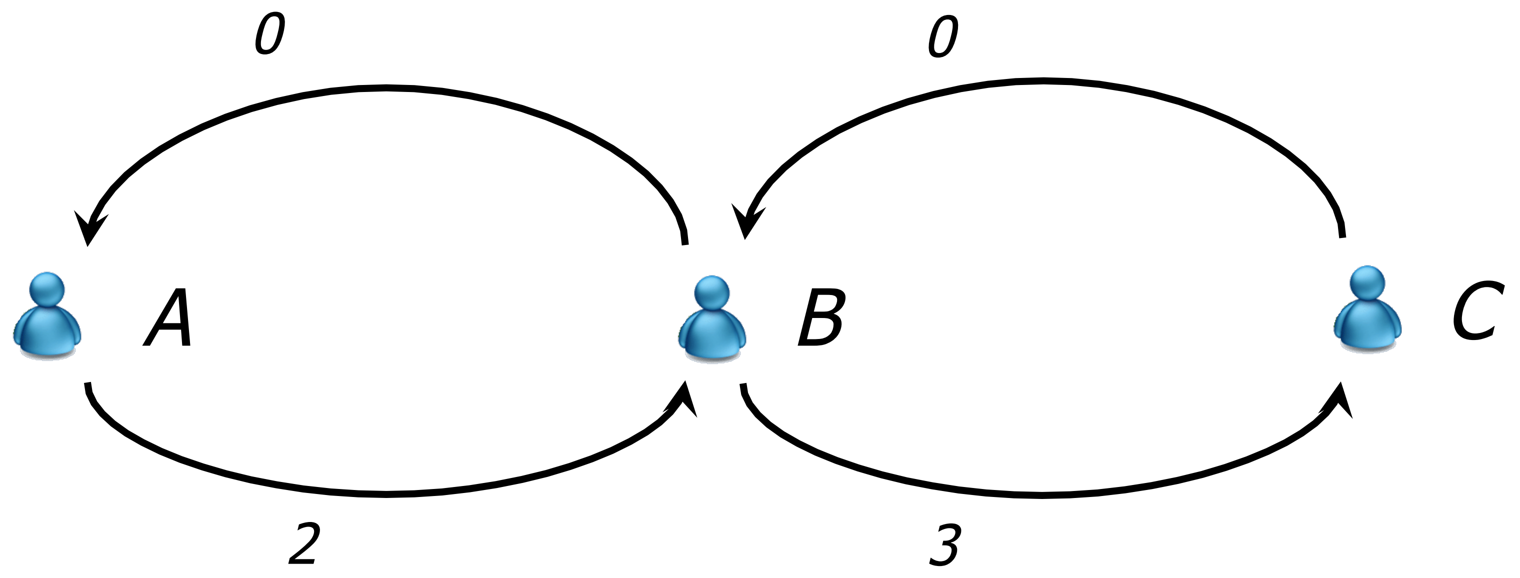
- Does not seek to identify Sybils
- Instead, limits the impact of Sybils on relevant system properties
- Examples:
  - Limit the amount of spam in an email system
  - Limit fraud in an online marketplace
  - Limit large-scale data aggregation in OSNs
- Users get no benefit from using multiple ids
- Unlike Sybil detection, requires application-specific information

Outline

1. Social systems
2. Accountability for distributed systems
3. Leveraging social relationships
   - Exploiting social networks for Sybil tolerance
   - Credit networks
   - Ostra: thwarting spam
   - Bazaar: limiting auction fraud
   - Genie: limiting social network crawling

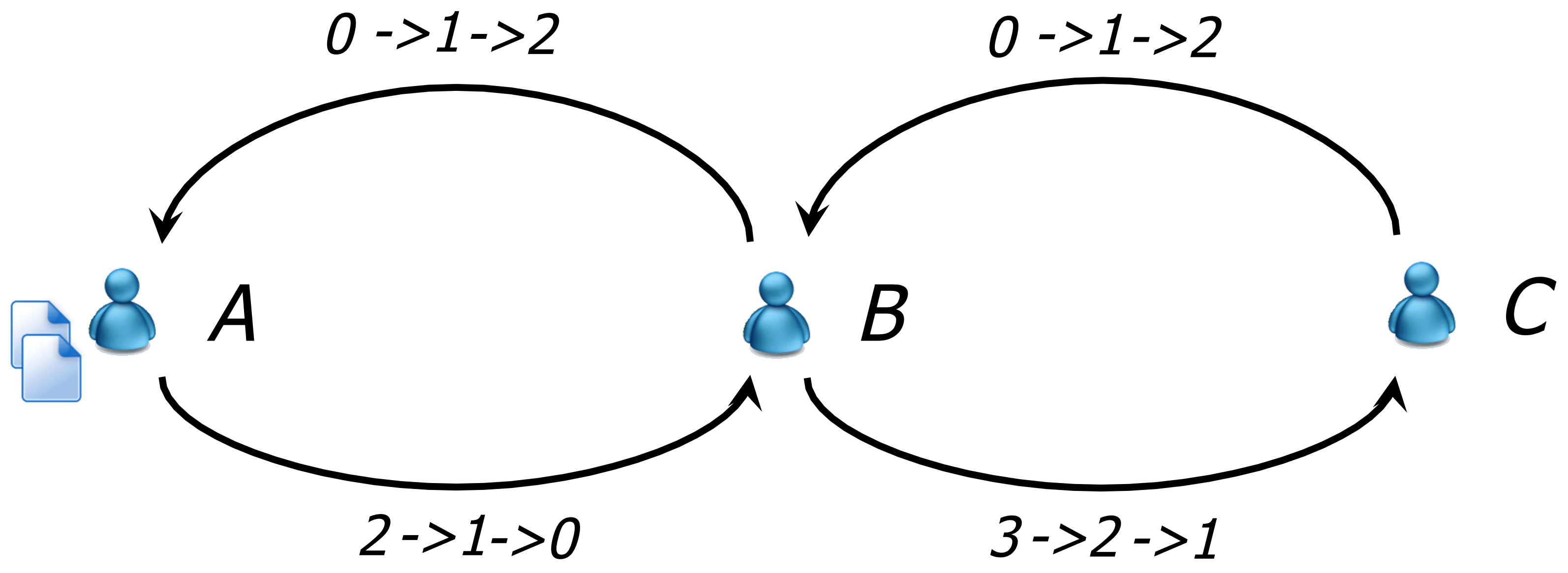
Credit network [*DeFigueiredo, CEC'05*], [*Ghosh, WINE'07*]

- Decentralized payment infrastructure (no common currency)
- Nodes form a directed graph
- Edge *(u,v)* with weight *n* means
  - *u* trusts *v* for up to *n* units of credit, or
  - *u* owes *v n* units of credit.



Credit network: Purchases

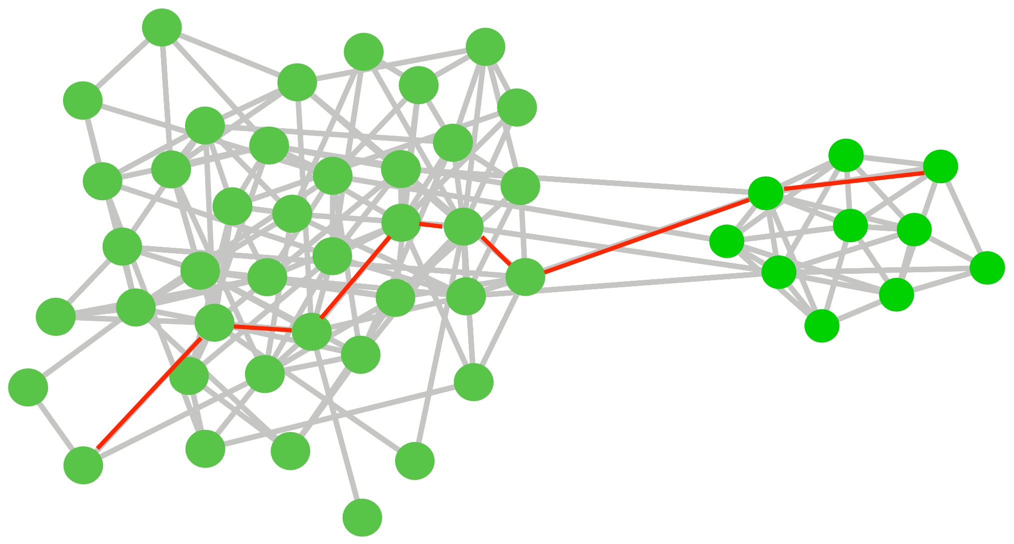
Node *w* can purchase goods or services worth *n* credits from *u* if every edge along a path *u* -> *w* has a weight >= *n*



What about intermediate node B?

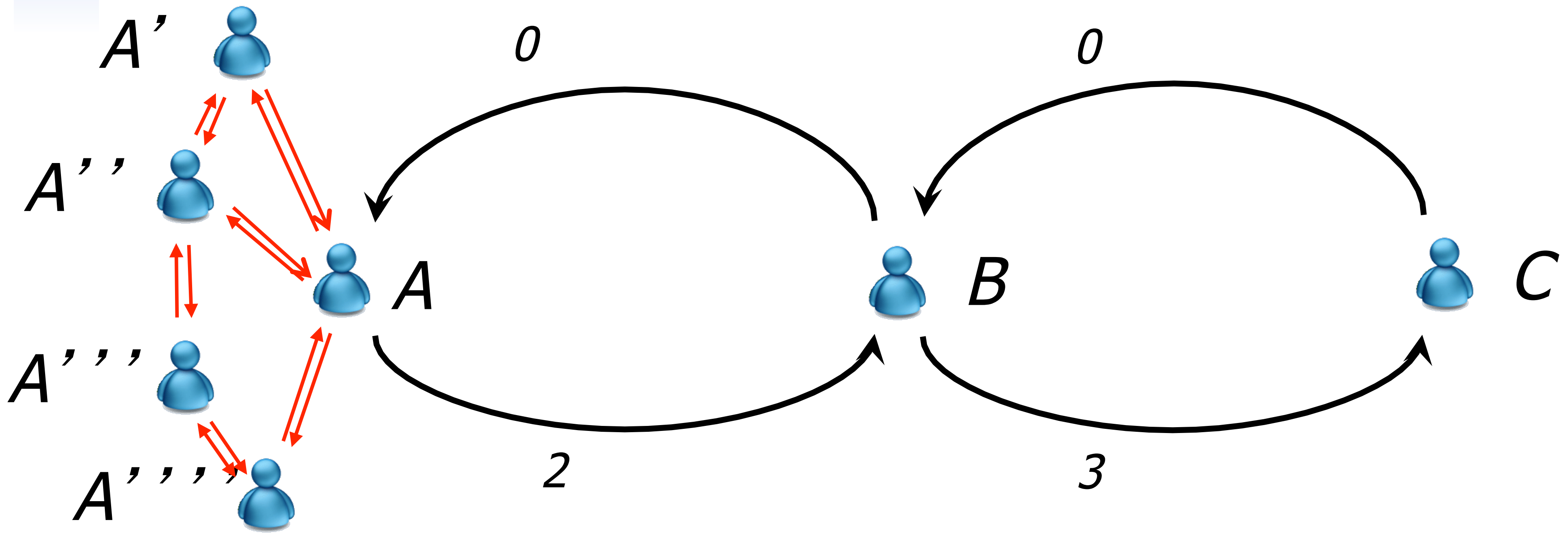
Its has lost two credits with C and gained two credits with A!

Credit network: Liquidity



- Intermediate nodes of a transaction are unaffected if the graph is richly connected
- Can use credit on a different path
- Under certain conditions, no significant loss of liquidity compared to a centralized payment infrastructure [*Dandekar, EC'11*]

Credit networks: Sybil tolerance



- Credit available to a Sybil attacker is limited by edges shared with real users
- Independently of how many ids the attacker controls!

Sybil-tolerant systems based on credit networks: Approach

1. Construct a credit network based on a social network
   - Credit network inherits SN's rich connectivity
   - Edges shared with real users limited by social capital

2. Map the desired Sybil-tolerant system property to an equilibrium ("trade balance $\pm\ \varepsilon$") or a max-flow in the credit network, using
   - Initial per-link credit assignment
   - Per-transaction payments
   - Credit adjustment

Example: P2p fair exchange

- Goal: Prevent freeloading in a p2p content sharing network

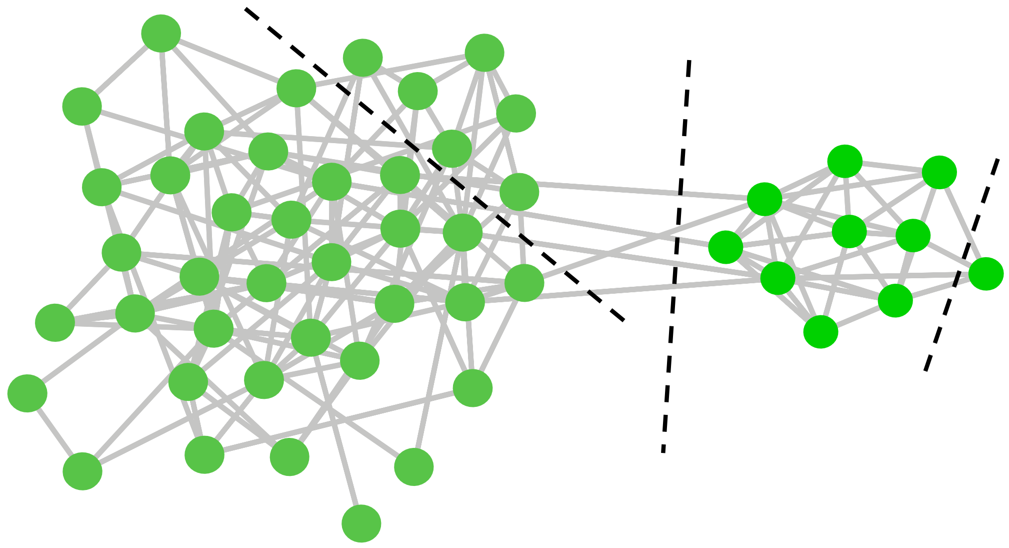
- Strawman solution: Trusted centralized account
  - Account keeps balance for every node:

    balance = initial credit + upload – download
  - Prone to Sybil attacks

    Attacker can boost balance by reporting fictitious uploads among Sybils

Sybil-tolerant p2p fair exchange

- Peers connected by a social network (SN)
- Downloader pays uploader 1 credit/block
  - using a credit path connecting them in the SN
- A node can download at most as much as it uploads
- Same holds for each connected component
  - including a component of Sybil nodes



Trade balance enforced along each edge cut

- - - - - - - - -

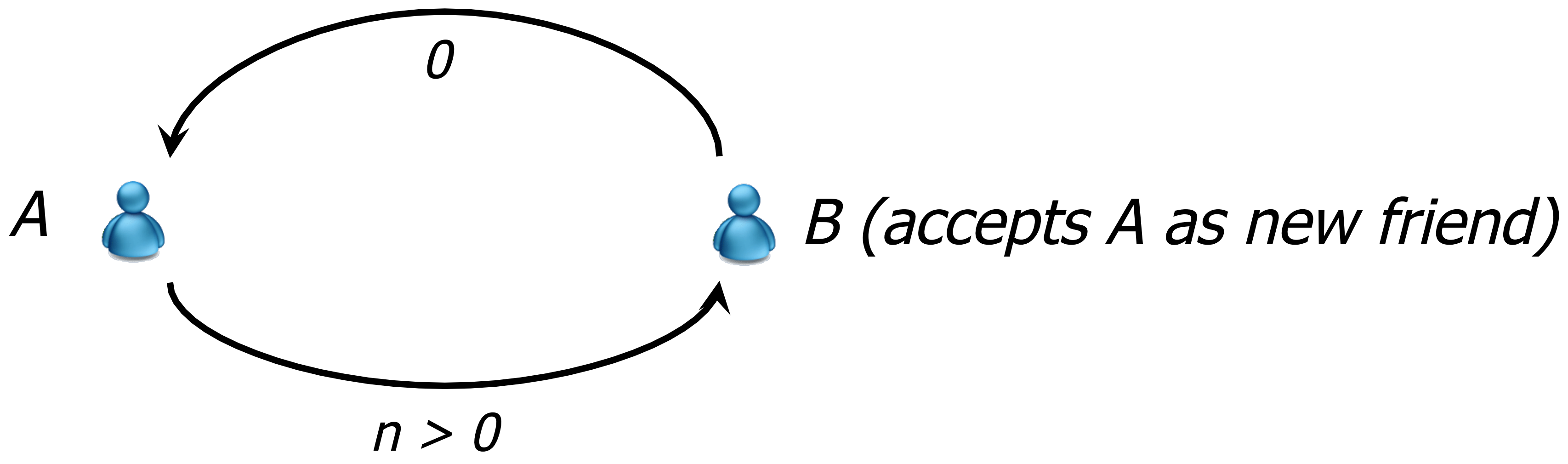
Questions

- How do we get started?
  - Friends grant each other some initial credit
- What about temporary “trade imbalances” ?
  - Initial credit provides a buffer
- How much initial credit is needed?
  - Can be determined experimentally
  - Increase until most nodes get what they want most of the time

Limiting initial credit

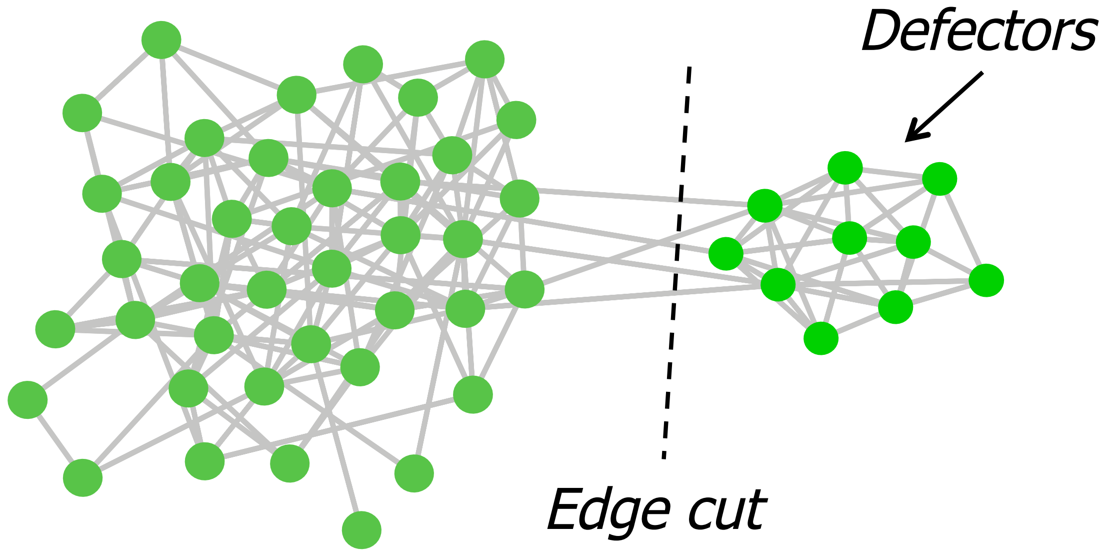
How do we prevent abuse of the initial credit?

- Limited problem, it is difficult to make new friends
- In addition, can require initiator to grant initial credit, while acceptor grants no initial credit



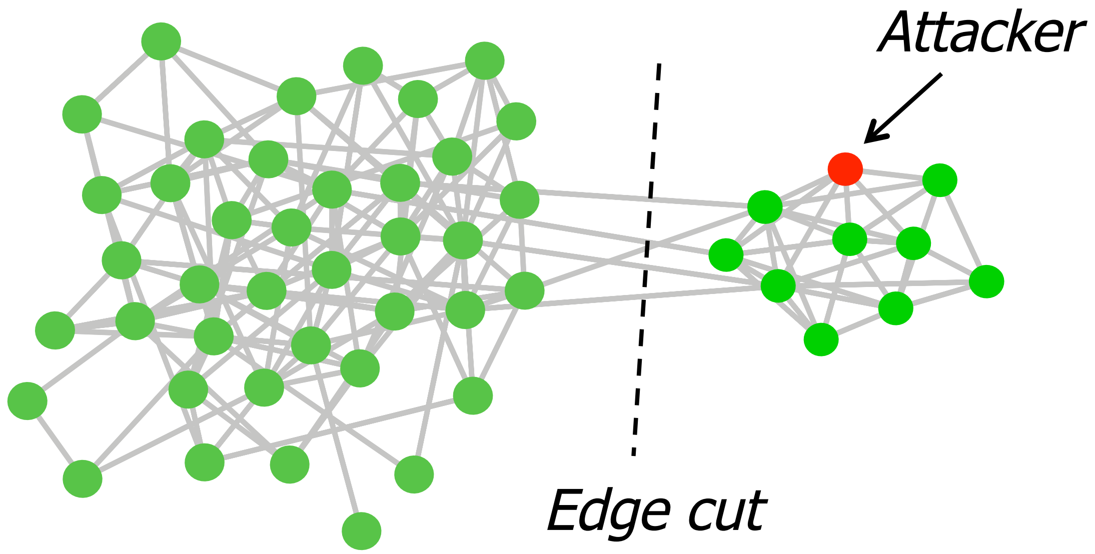
Effects of defection

- What is the damage when a (group of) node(s) defects?
- Size of edge cut times initial credit granted to defectors

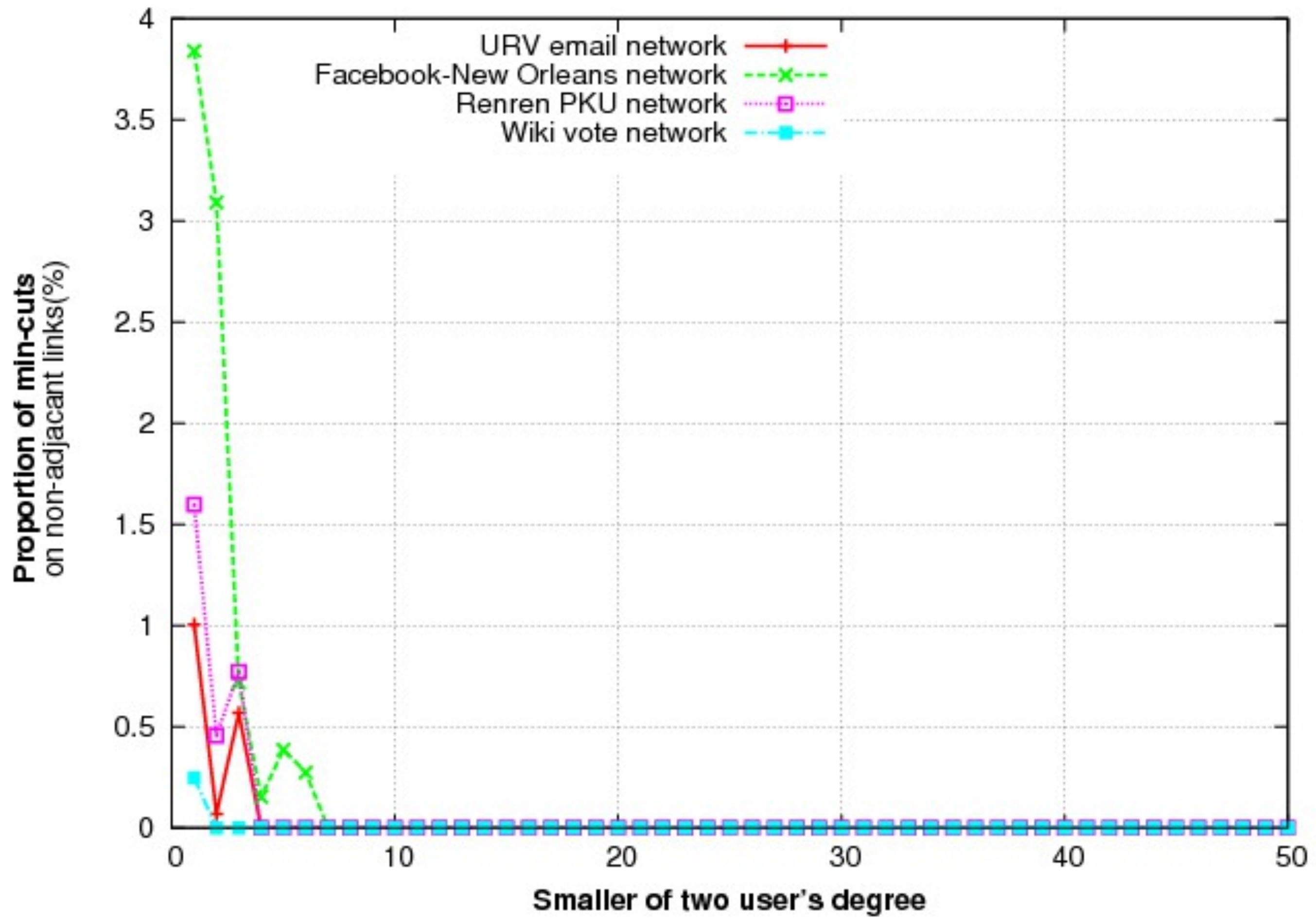


Denial of service

- Can a (group of) node(s) deny service to unrelated good nodes?
- By exhausting credit along a cut between good nodes!
- Yes, but the attacker must control an edge cut larger than the victim's cut to the rest of the network

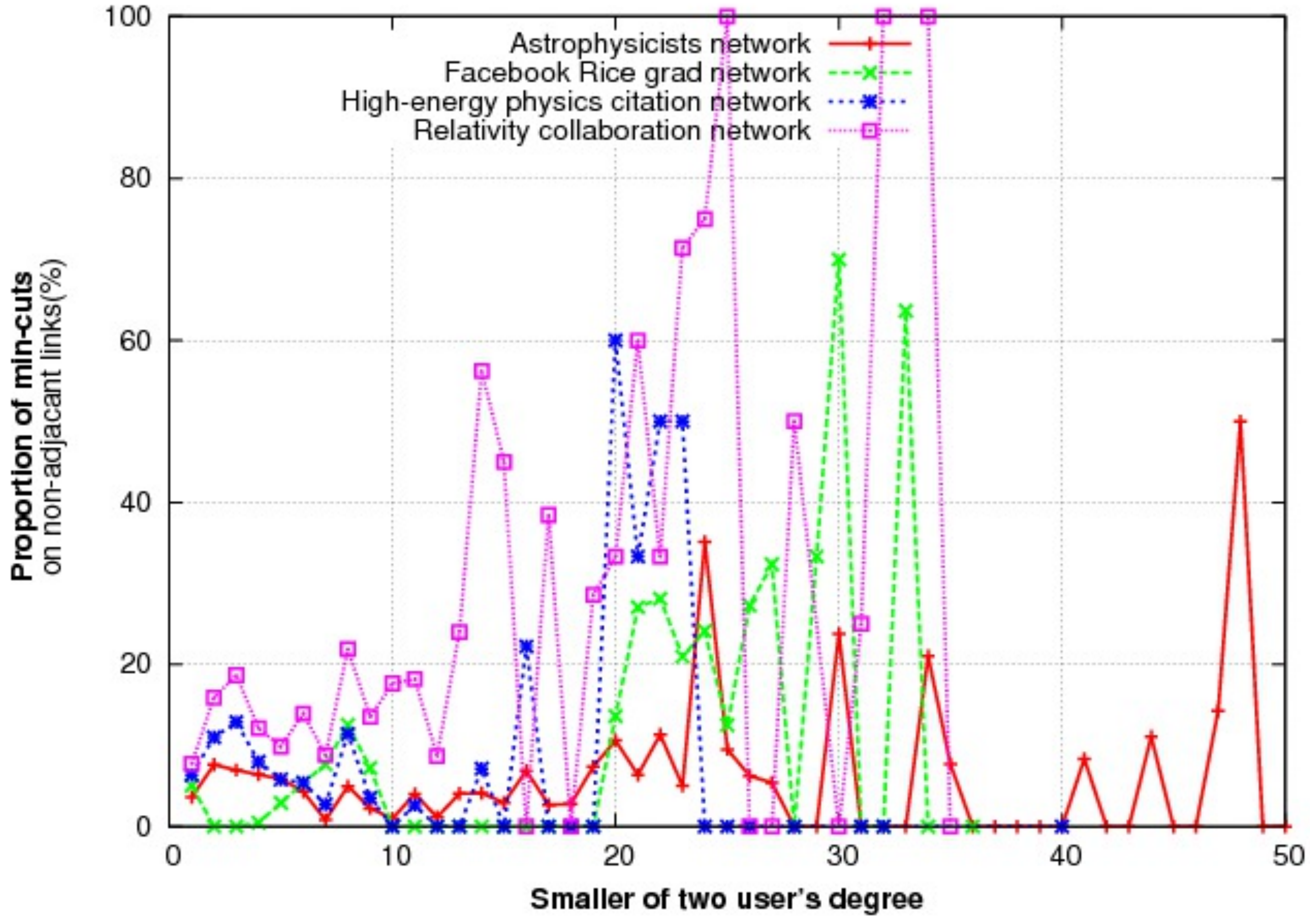


Potential for credit DoS: SNs with modularity < .6



- 3000 randomly selected pairs of users
- Only low-degree nodes are vulnerable

Potential for credit DoS: SNs with modularity > .6



- 3000 randomly selected pairs of users
- Nodes with degree < 50 are vulnerable

Outline

1. Social systems
2. Accountability for distributed systems
3. Leveraging social relationships
   - Exploiting social networks for Sybil tolerance
   - Credit networks
     - Ostra: thwarting spam
     - Bazaar: limiting auction fraud
     - Genie: limiting social network crawling

Ostra [Mislove NSDI' 08]: Thwarting unwanted communication

- Email spam
- Search-engine spam
- Mislabeled content on sharing sites
- Unwanted invitations in Skype, IM systems

- Existing solutions expensive
  - Costly arms race
  - Content filtering for rich media?

Ostra: Assumptions

- Users form a social network, e.g.,
  - People who appear in each other's contact list
  - People who have ever exchanged email
- Receivers classify content
  - Explicit (Junk button)
  - Implicit (Deletion, lack of response)

Ostra: Mechanism

- **Network:**
  - Directed link pair between prior correspondents
  - Link weight is balance of spam from either side
  - Initial condition: Per-link credit of 3 both ways
- **Equilibrium:**

  rate of spam received ≈ rate of spam sent along each cut
- **Transaction:** unwanted communication received

  unwanted communication incurs a charge of 1 credit from sender to receiver
- **Credit adjustment:**

  Credit balances decay at constant rate (e.g., 10%/day)

Ostra: Results

- Rate of spam a (group of) nodes can send is proportional to number of links they have:

  (decay * #links) + rate of received spam

- Evaluated on real social network and email traffic data
  - Rate of spam with 20% attackers is

    4 msgs/user/week (Initial credit of 3; 10% decay per day)
  - 1.3% of messages are delayed by a few hours

Outline

1. Social systems
2. Accountability for distributed systems
3. Leveraging social relationships
   - Exploiting social networks for Sybil tolerance
   - Credit networks
   - Ostra: thwarting spam
   - Bazaar: limiting auction fraud
   - Genie: limiting social network crawling

Bazaar [Mislove NSDI' 11]: Limiting auction fraud

- Online marketplaces suffer from fraud
- Ebay, Overstock, uBid, Amazon Marketplace
- User identities and reputations are subject to
  - Whitewashing
  - Collusion
- Significant damage
  - Recent convict defrauded US$ 717k from 5000 eBay users using > 250 eBay accounts

Bazaar: Mechanism

- **Network:**
  - Undirected link between pairs of users who have transacted
  - Link weight is balance of values of transactions with + and – feedback
  - Initial condition: OSN friends trust each other with initial amount; or, value of amount placed in escrow
- **Condition:**

  Value of new transaction must not exceed max-flow between seller and buyer
- **Transaction:** (k = transaction value)
  - + feedback: increase path weights by k
  - - feedback: decrease path weights by k
  - Neutral feedback: no change

Bazaar: Results

- 90-day UK eBay trace, over 3M users
  - Over 8M transactions with buyer feedback
- Would have flagged GBP 164k worth of negative feedback transactions (36% of transactions with negative feedback)
- False positive rate of less than 5% (transactions flagged by Bazaar that resulted in positive feedback)

Outline

1. Social systems
2. Accountability for distributed systems
3. Leveraging social relationships
   - Exploiting social networks for Sybil tolerance
   - Credit networks
   - Ostra: thwarting spam
   - Bazaar: limiting auction fraud
   - Genie: limiting social network crawling

Genie: Limiting OSN crawls

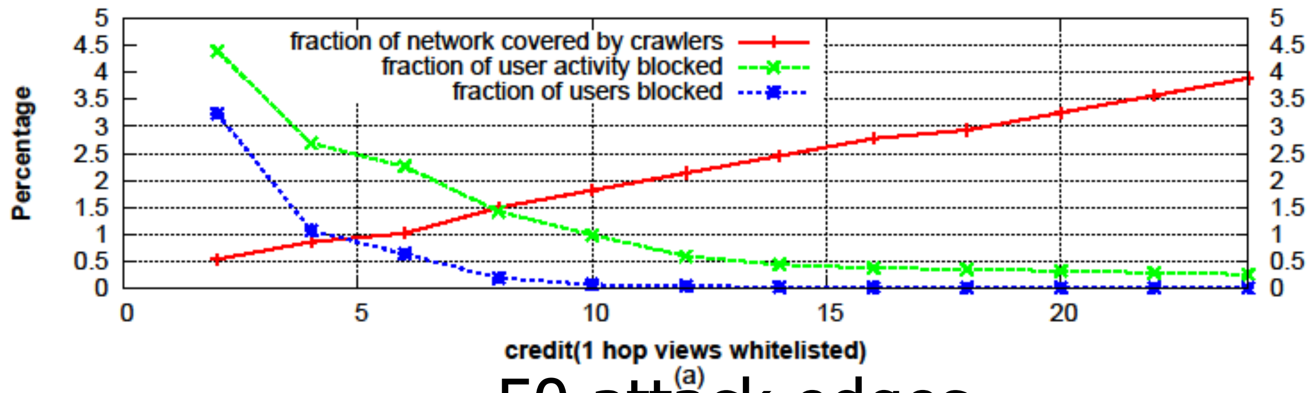
- OSN users and operators wish to limit data aggregation by crawlers
  - Users wish to limit their exposure
  - Operators wish to protect their data assets
- Existing rate-limiting techniques are vulnerable to Sybil attack
- E.g.: Someone crawled over 100k public Facebook user profiles and shared them via BitTorrent

Genie: Mechanism

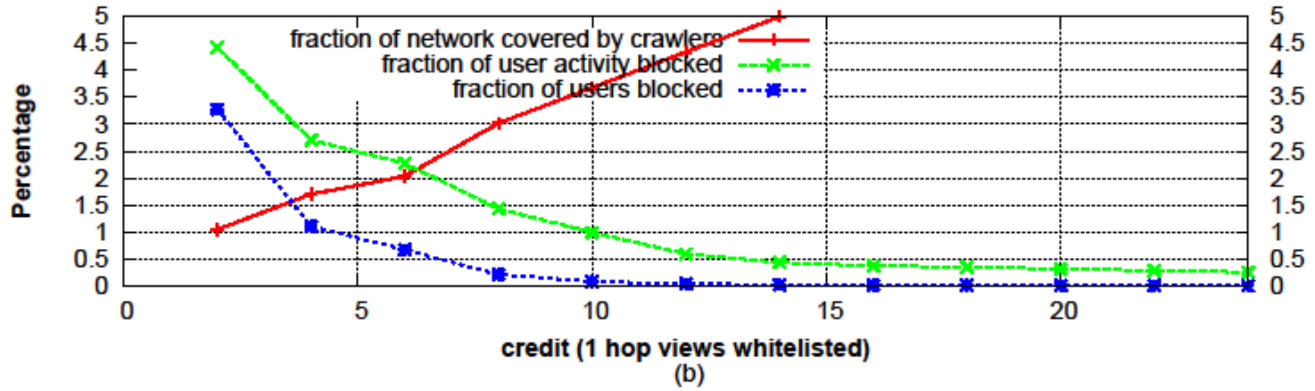
- **Network:**
  - Directed link pair between OSN friends
  - Link weight is balance of views from either side
  - Initial condition: Link requestor grants acceptor 1 credit
- **Equilibrium:**
  - Number of views from either side of a cut is roughly balanced
- **Transaction:** Viewer views viewee's profile
  - Viewer->Viewee -1; Viewee->Viewer +1
  - One-hop views are free
- **Credit Adjustment:**
  - Link credit is rebalanced at a rate of 5 views / week

Genie: Results

2-week trace of profile views from a sample of the Renren OSN (705k links, 33k nodes – 0.02% of nodes)



50 attack edges



100 attack edges

Other Sybil tolerant applications using trust networks

- SumUp [*Tran et al. NSDI'09*]
  - Sybil-tolerant voting

- [*Levien USENIX Sec'98*], [*Cheng P2PECON'05*], [*Ziegler Inf. Sys. Frontiers'05*]
  - Reputation systems

Challenges

Credit networks enable Sybil-tolerant system properties

- Systems can use weak identities
- Users can use multiple pseudonyms

But…

- Mechanism design is application-specific
- Need to understand scope of possible applications
- Systematic design of solutions
- Need to understand social network structure

Challenges

- Need to understand how social network and credit network shape each other dynamically (social dynamics)

- Social pressure shapes the network
- Credit network incentivizes users to chose wisely who they wish to associate with
- Drop a friend whose requests have repeatedly caused me to lose liquidity

Conclusion

- Effectiveness of Sybil detection on real social networks remains unclear
- Sybil tolerance instead seeks to make a system's properties of interest robust to Sybils
- Credit networks provide a general model for Sybil tolerant systems
- Current point solutions cannot easily be generalized
- Need to understand the social dynamics

Max Planck Institute for Software Systems (MPI-SWS)

- Government-funded basic research institute, since 2005
- Combines best of academia and research lab
  - Academic freedom, doctoral students and post-docs
  - Generous funding and excellent facilities
- Up to 18 (currently 11) faculty members / groups
- Highly international, working language is English
- Graduate school
- Opportunities for outstanding doctoral students and post-docs; also, positions for visiting faculty

http://www.mpi-sws.org

MPI-SWS Faculty

- Umut Acar (CMU 2005): Programming Systems
- Björn Brandenburg (UNC 2011): Real-time Systems
- Paul Francis (UCL 1994): Internet architecture, privacy
- Derek Dreyer (CMU 2005): Programming Languages
- Peter Druschel (Arizona 1994): Distributed Systems
- Deepak Garg (CMU 2009): Security and Privacy
- Krishna Gummadi (UW 2005): Social networks/systems
- Rupak Majumdar (Berkeley 2003): Software Verification
- Ruzica Piskac (EPFL 2011): Automated Reasoning
- Rodrigo Rodrigues (MIT 2005): Dependable Systems
- Viktor Vafeiadis (Cambridge 2010): Logic, Verification

Thanks for your attention!