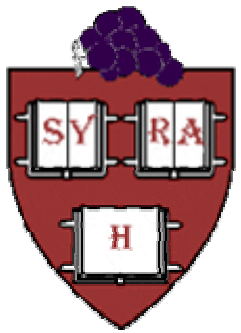
Strategyproof Computing: Systems Infrastructures for Self-Interested Parties

Chaki Ng, David C. Parkes, Margo Seltzer
Harvard University



Workshop on Economics of Peer-to-Peer Systems, June 5-6, 2003, Berkeley, CA

Outline

- Introduction
- Motivations and Goals
- Economic Foundations and Components
- Validation Proof-of-Concept
- Challenges

Vision of Strategyproof Computing

- Open systems in which
  - Resource allocation and negotiation schemes are strategyproof
  - Parties are freed from game-theoretic reasoning
  - Can treat other resources as their own
- Advocate that systems be
  - Populated with strategyproof mechanisms that are deployed by resource owners and intermediaries
  - Supported by a lightweight “dialtone” that certifies and validates
- Infrastructure agenda
  - Build out of multiple mechanisms

Terms

- Resource – computing resources in a P2P systems
  - Examples: CPU, storage, bandwidth, applications, services
- System – computing environment in which
  - Multiple resources are deployed
  - Examples: p2p, grid, Internet
- Infrastructure – the glue in the system that
  - Ties the resources together
- Example: for a P2P wireless system
  - Resources: location-based services, data staging, sensor data
  - Infrastructure: mobile IP, wireless protocols like bluetooth

Outline

- Introduction
- Motivations and Goals
- Economic Foundations and Components
- Validation Proof-of-Concept
- Challenges

Motivations

- P2P systems
  - Parties are self-interested
  - Resources are owned and used by different parties
  - Resources and requirements change dynamically
- What are the problems?
  - Parties will strategize, for benefits or for defense
    - Example: P2P message routing
  - Strategic behaviors add complexity to P2P systems
    - Costly for systems as a whole – hard to "fix"
    - Costly for individual parties
      - Computational constraints
      - Uneven abilities to strategize

"Embrace and simplify self-interest in P2P systems."

Goals I

"Build systems where the optimal strategy for a self-interested party is simple."

- Incentives-first
  - Embrace self-interest explicitly, much like fault-tolerance and security
- Utility-based
  - Model and maximize utilities of parties in the systems
  - Link utilities to common currencies
- Simple
  - Ease expensive frequent gaming
  - Parties can participate with simple strategies

Goals II

"Enable open infrastructures that support decentralized deployment of resources."

- Open
  - Create the "dialtone" for scalable systems
  - Enable innovation and competition
- Decentralized
  - No one single mechanism will work
  - Let users specify and deploy with "limited scope"

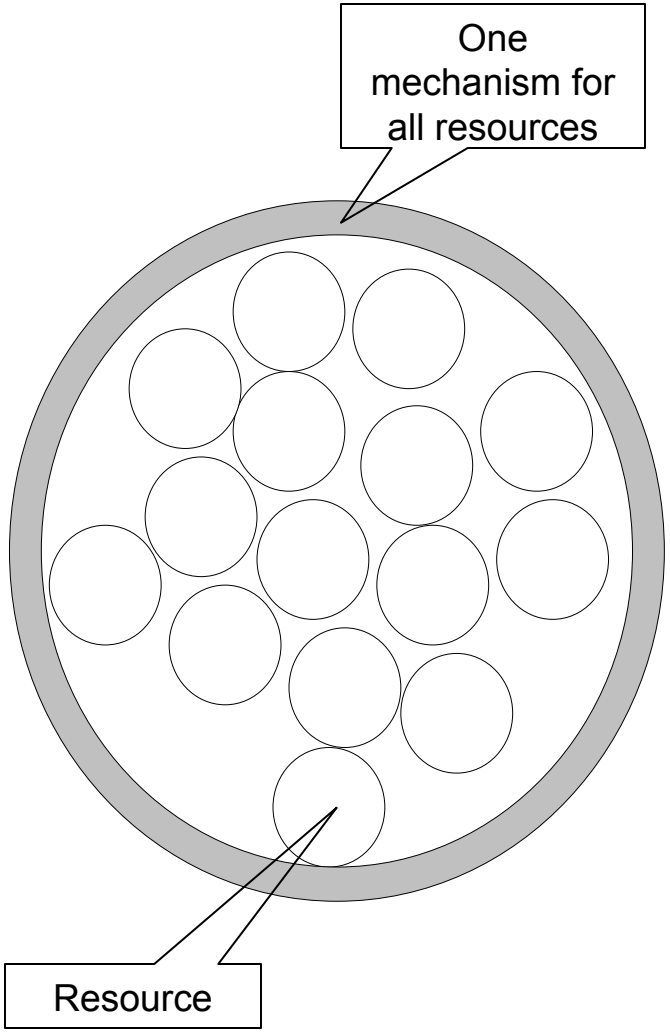
Outline

- Introduction
- Motivations and Goals
- Economic Foundations and Components
- Validation Proof-of-Concept
- Challenges

Mechanism Design

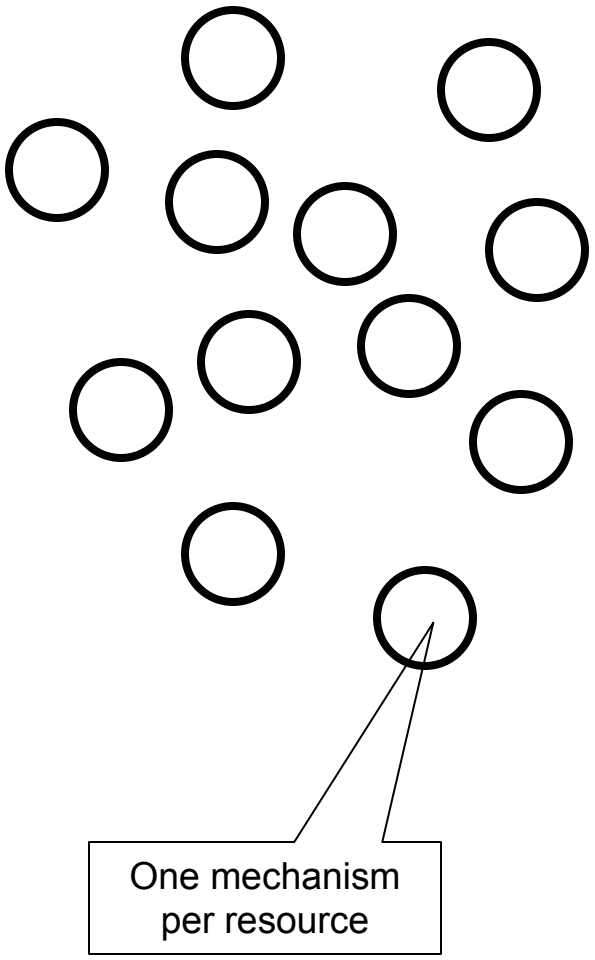
- Mechanism design
  - Design “rules of the games,” for parties with private preferences
  - Implement output specifications and payments
  - Consider constraints in participation, incentive-compatibility, and budget-balanced
  - Choose among desirable properties like efficiency, fairness, or revenue maximization.
- Example
  - Auctions: award item to highest-value bidder
- A strategyproof mechanism is one in which simple truth-telling is the dominant strategy for a self-interested party
  - Example: second-price auction
  - First goal met

One Strategyproof Mechanism?



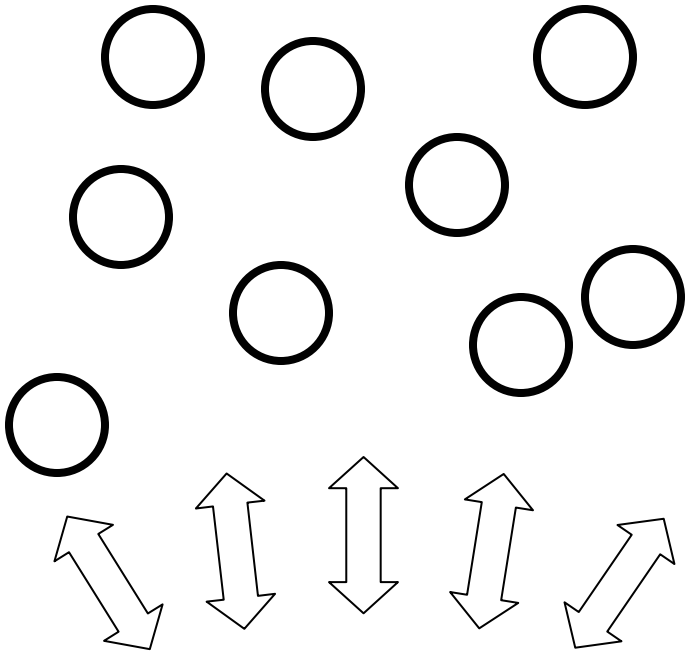
- Traditional MD assumes a single global solution for all resources in a system
- Issues
  - Timeliness of information
  - Computation
  - Authority issues and trust
  - Unforeseen new resources
- Combinatorial auctions do not help much
  - Cannot build one combinatorial auction for the entire world

Totally Decentralized?



- Break things up
  - Now things are decentralized!
- Give up
  - Global strategyproofness
- Get
  - Market of “locally-strategyproof” mechanisms
  - Local – within limited resource space a mechanism provides
- Problems remain…
  - Cannot ensure mechanisms are strategyproof…

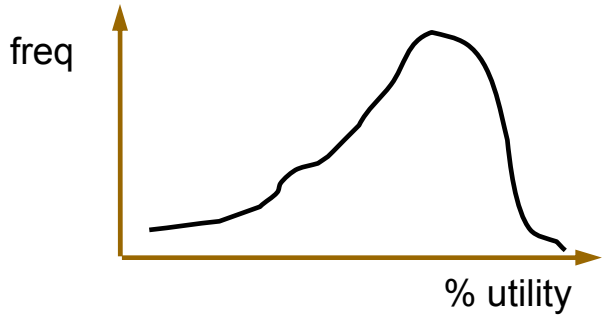
Our World



- Enforce strategyproofness
  - Add infrastructure
    - Certification
    - Validation
- Both goals matched
- Scope matches user needs
  - Geographic
  - Organizational
- “Right” scope
  - Emerge by competition

Components

- A strategyproof computing system includes
  1. Locally-strategyproof mechanisms
  2. Infrastructure
  3. Interface provided by each mechanism
     - Request language ("what can you tell me?")
       - Bid on discrete items, vs. bundle of items
     - Type assumption ("what do I assume about you?")
       - Strategyproof for defined types
     - Statistics ("what surplus do I generate?")



Some of Many Key Challenges

- Composition
  - Multiple mechanisms are not together strategyproof
- Building
  - How do developers build and test mechanisms?
- Validation
  - How to validate locally-strategyproof mechanisms?

Outline

- Introduction
- Motivations and Goals
- Economic Foundations and Components
- Validation Proof-of-Concept
- Challenges

Validation

- Mechanisms
  - Get certificates
  - Present to users
  - Maintain status for validation
- Possible implementations
  - Active monitoring: “police” agents
  - Passive monitoring: audit request and result info
- Novelty: do not want to know mechanism algorithms
  - Keep things open and decentralized

Validation: Proof-of-Concept

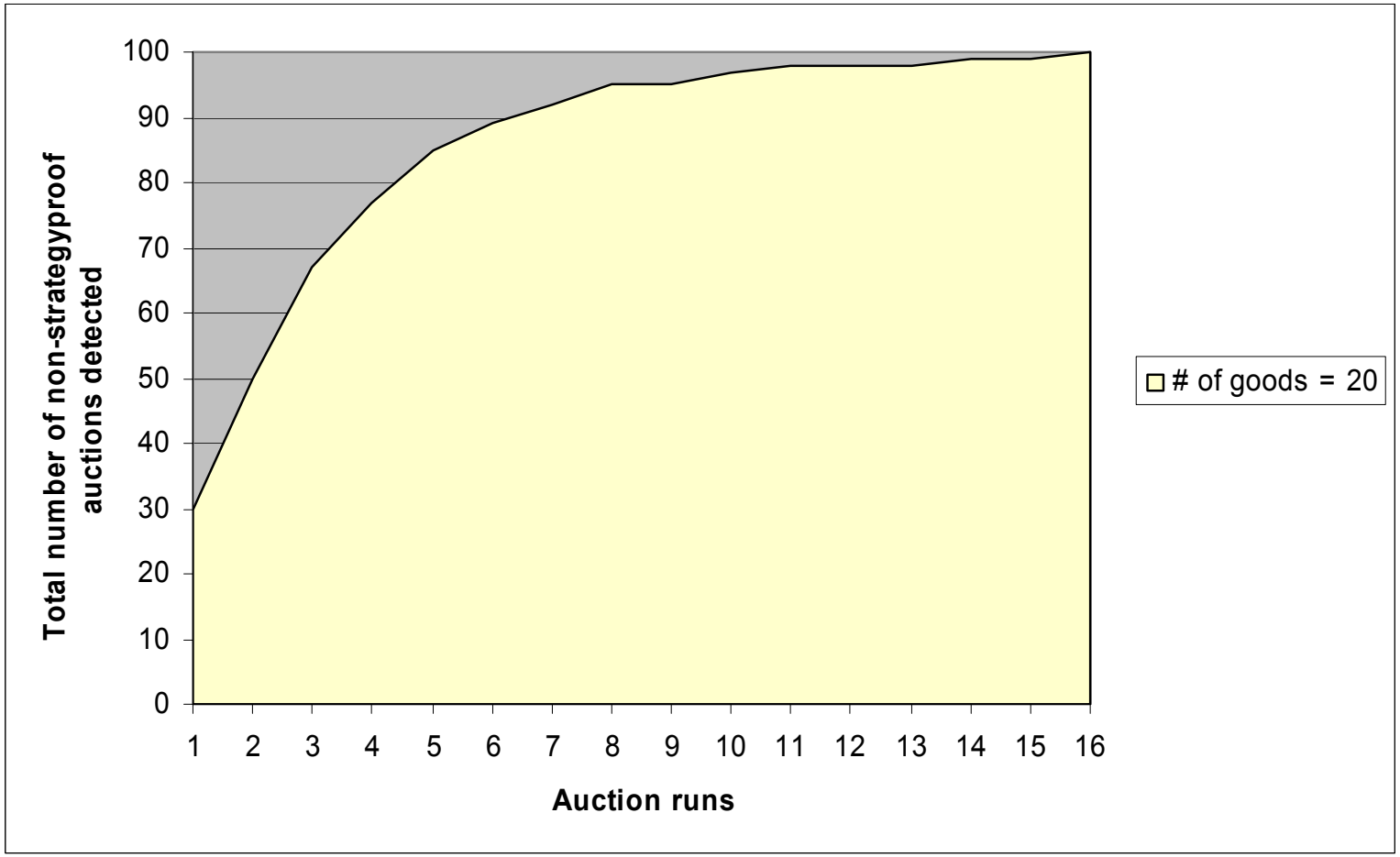
"How quickly can we detect
non-strategyproof mechanisms?"

- Passive monitoring
  - Validation - observes the bids and the outcomes
- Consider first-price sealed bid auction (FPSB), non-strategyproof
  - Incentives to bid lower than true value
- Necessary condition for strategyproofness
  - Utility(my bid, my outcome) >= Utility(my bid, any other outcome)
- Non-strategyproof example
  - FPSB with two units, two bidders
  - I bid $10 for one unit, someone bids $8 for one unit
  - Bad! Should bid $8 instead

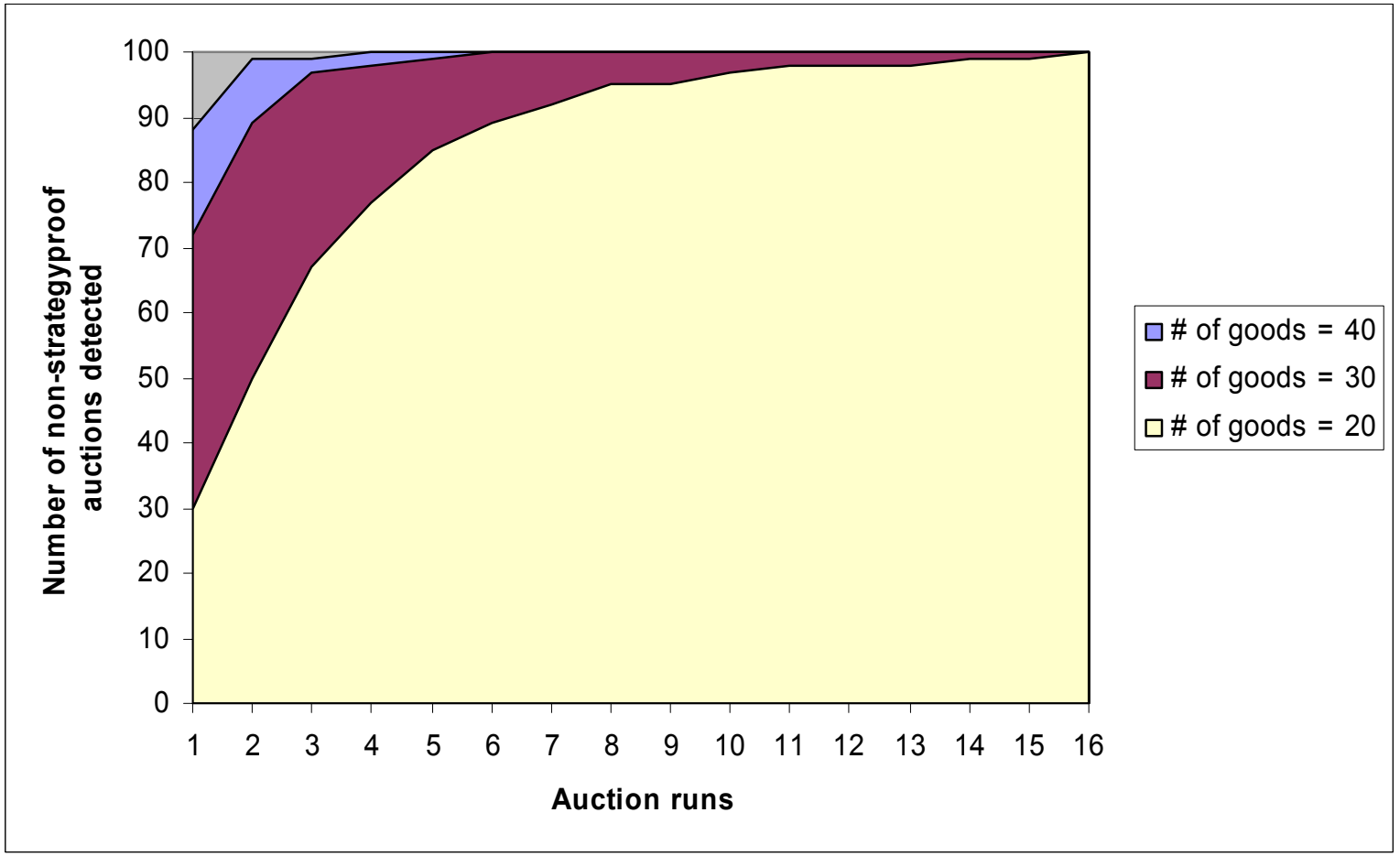
Validation: Simulations

- Multi-unit mechanism, FPSB
  - Sells same number of identical goods
  - Parties submit a value for number of units desired; want all-or-nothing
- Mechanism runs over time
  - For each run, assume fixed number of parties bidding (e.g. 20)
- Weighted random distribution
  - Parties' units: draw uniformly from (1, 10)
  - Parties' value: draw uniformly from (1, units)
- Data
  - Note during which run non-strategyproofness is detected
  - Run the mechanism 100 times

Validation: Results



Validation: Results



Outline

- Introduction
- Motivations and Goals
- Economic Foundations and Components
- Validation Proof-of-Concept
- Challenges

Challenges

- How to implement strategyproof mechanisms?
  - Development and debugging tools
  - Library of mechanisms: decentralized, online, two-sided exchanges
  - SP limiting - approximately-LSP necessary?
- Validation
  - Police agents: how not to distort mechanisms?
  - Passive: How much data to be kept around?
  - Catching short-lived mechanisms: evaluation period?
- Market of Mechanisms
  - Possible evolutions
  - Composition among mechanisms
- Measurements
  - What and how to measure complexity?
  - How users model their utilities?

Summary

- Strategyproof computing is
  - A unified way for users to perform direct, simple computing
  - An infrastructure effort for heterogeneous strategyproof mechanisms
- And is not
  - For systems where a single mechanism may suffice
  - About designing (but adopting and building) strategyproof mechanisms

chaki@eecs.harvard.edu

http://www.eecs.harvard.edu/EconCS

http://www.eecs.harvard.edu/SYRAH