

A Distributed Platform for Mechanism Design

Krzysztof R. Apt

CWI and University of Amsterdam

(joint work with Farhad Arbab and Huiye Ma)

Intelligent Design

Economist.com Search Economist.com Log in: e-mail Password 12 weeks for €26 [SUBSCRIBE NOW >>](#)

Requires subscription Remember me Register

Finance & Economics

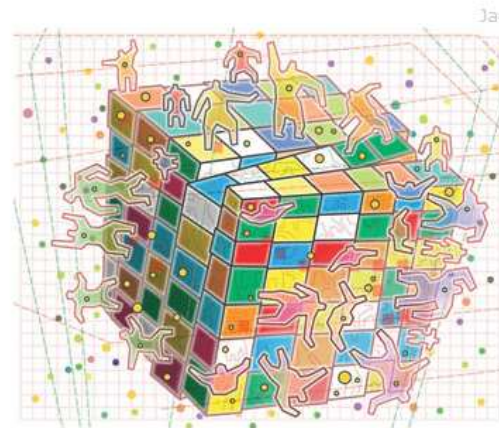
Economics focus

Intelligent design

Oct 18th 2007

From *The Economist* print edition

A theory of an intelligently guided invisible hand wins the Nobel prize



"WHAT on earth is mechanism design?" was the typical reaction to this year's Nobel prize in economics, announced on October 15th. In this era of "Freakonomics", in which everyone is discovering their inner economist, economics has become unexpectedly sexy. So what possessed the Nobel committee to honour a subject that sounds so thoroughly dismal? Why didn't they follow the lead of the peace-prize judges, who know not to let technicalities about being true to the meaning of the award get in the way of good headlines?

In fact, despite its dreary name, mechanism design is a hugely important area of economics, and underpins much of what dismal scientists do today. It goes to the heart of one of the biggest challenges in economics: how to arrange our economic interactions so that, when everyone behaves in a self-interested manner, the result is something we all like. The word "mechanism" refers to the institutions and the rules of the game that govern our economic activities, which can range from a Ministry of Planning in a command economy to the internal organisation of a company to trading in a market.

Intelligent Design

A theory of an intelligently guided invisible hand wins the Nobel prize

WHAT on earth is **mechanism design**? was the typical reaction to this year's Nobel prize in economics, announced on October 15th.

[...]

In fact, despite its dreary name, mechanism design is a hugely important area of economics, and underpins much of what dismal scientists do today. It goes to the heart of one of the biggest challenges in economics: how to arrange our economic interactions so that, **when everyone behaves in a self-interested manner, the result is something we all like.**

(The Economist, Oct. 18th, 2007)

Executive Summary

- We describe design of a structured, highly flexible platform for **distributed mechanism design**.
- The system is built as a sequence of **layers**.
- **Lower** layers deal with the operations relevant for distributed computing.
- **Upper** layers deal with the relevant aspects of the mechanism design.
- Specific applications are realized as instances of the **top layer**.
- The implementation supports various instances of mechanisms.

Decisions, Decisions, . . .

Assume

- **players** $1, \dots, n$,
- set of **decisions** D ,
- for each player
 - set of **types** (e.g., **valuations**) Θ_i ,
 - utility function

$$v_i : D \times \Theta_i \rightarrow \mathcal{R}$$

that he wants to maximize

(**behaves in a self-interested manner**).

- **Decision rule**: a function $f : \Theta_1 \times \dots \times \Theta_n \rightarrow D$.

Mechanism Design: Classical View

The following sequence of events:

- each player i has type (e.g., valuation of an item) θ_i ,
- each player i announces to the central authority a type (e.g., a bid) θ'_i ,
- the central authority computes decision

$$d := f(\theta'_1, \dots, \theta'_n)$$

and communicates it to each player.

Problem to solve: Each player i wants to manipulate the choice of $d \in D$ so that his utility $v_i(d, \theta_i)$ is maximized.

Example: Public Project Problem (1)

- There are 4 companies.
- A **bridge** needs to be constructed.
- Each company submits to the central authority its **valuation** (willingness to pay).
- Bridge gets built when **total** exceeds 40.
- If yes, each company **pays** 10.

Public Project Problem: Manipulations

Each company wants to **manipulate** the choice so that its utility is maximized.

In the public project problem ($c = 40$, $n = 4$):

- if $\theta_i \geq \frac{c}{n}$, submit c ,
- if $\theta_i < \frac{c}{n}$, submit 0.

Tax-based Mechanisms

- The central authority computes
 - decision $d := f(\theta')$,
 - the sequence of **taxes** $(t_1, \dots, t_n) := g(\theta')$, and
 - communicates to each player i decision d **and** tax t_i he needs to pay (if $t_i < 0$) or receive (if $t_i \geq 0$),
- the resulting utility for player i :
 $u_i(d, t) := v_i(d, \theta_i) + t_i$.
- **Groves mechanisms**
By specific use of taxes (function g) **truth-telling** (reporting $\theta'_i := \theta_i$) becomes **best strategy** for player i .
(**Cheating does not pay off**).

Example: Public Project Problem (2)

- $d \in \{0, 1\}$,

- $t_i(\theta'_i, \theta_{-i}) =$

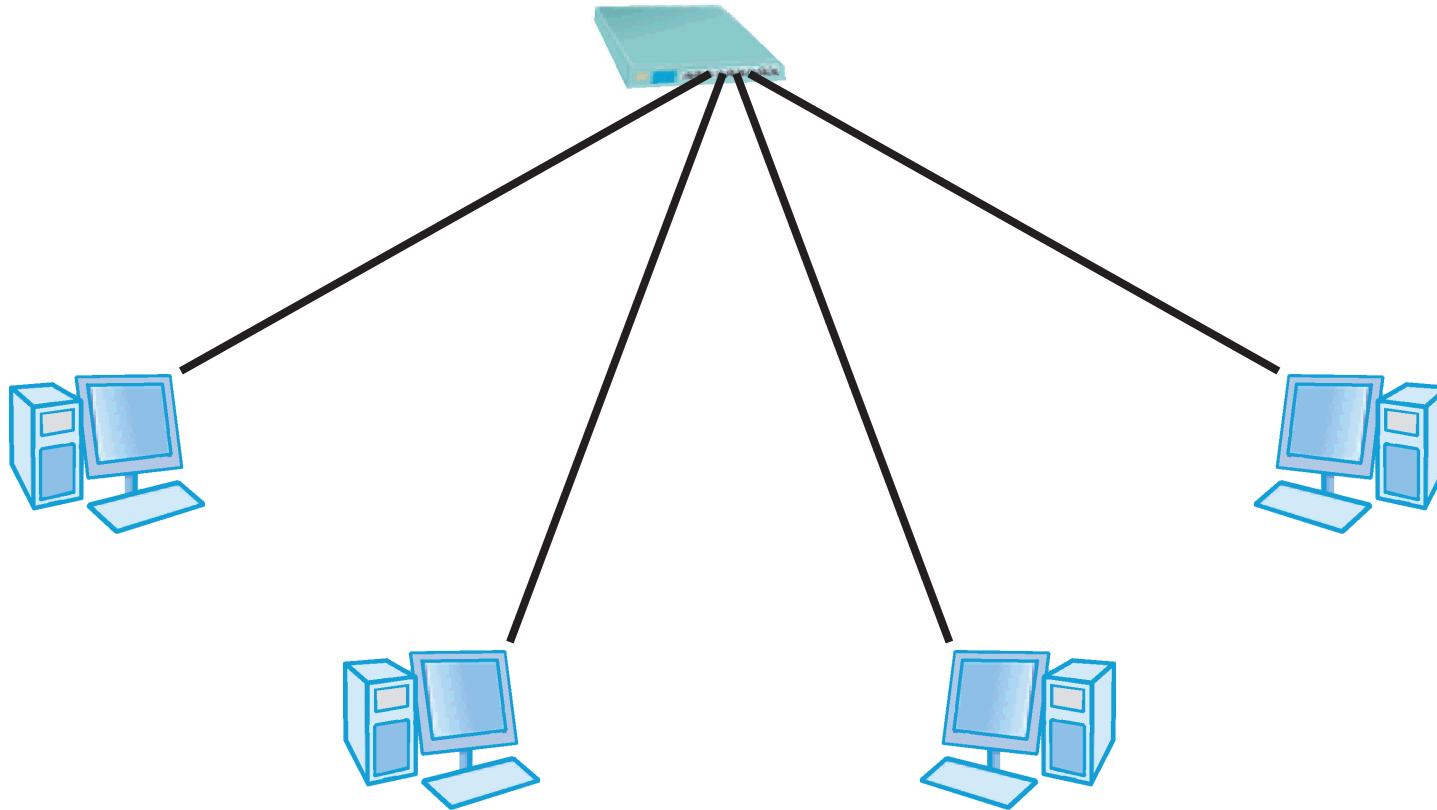
$$\begin{cases} \min(0, \frac{n-1}{n}c - \sum_{k \neq i} \theta_k) & \text{if } \sum_{k \neq i} \theta_k + \theta'_i < c \\ \min(0, \sum_{k \neq i} \theta_k - \frac{n-1}{n}c) & \text{otherwise} \end{cases}$$

Dept	type	submitted type	tax	util
A	15	15	0	0
B	11	11	0	0
C	11	11	0	0
D	5	0	-7	-7

Some Examples of Groves Mechanisms

- Vickrey auction.
Sealed bid auction.
The winner pays the **second highest bid**.
- Decisions concerning **public projects**.
- Various forms of **auctions**:
 - adwords (Google),
 - spectrum auction (Federal Communications Commission, FCC),
 - assigning takeoff and landing rights at the airports,
 - bus route auctions (London),
 - trucking services auctions,
 - ...

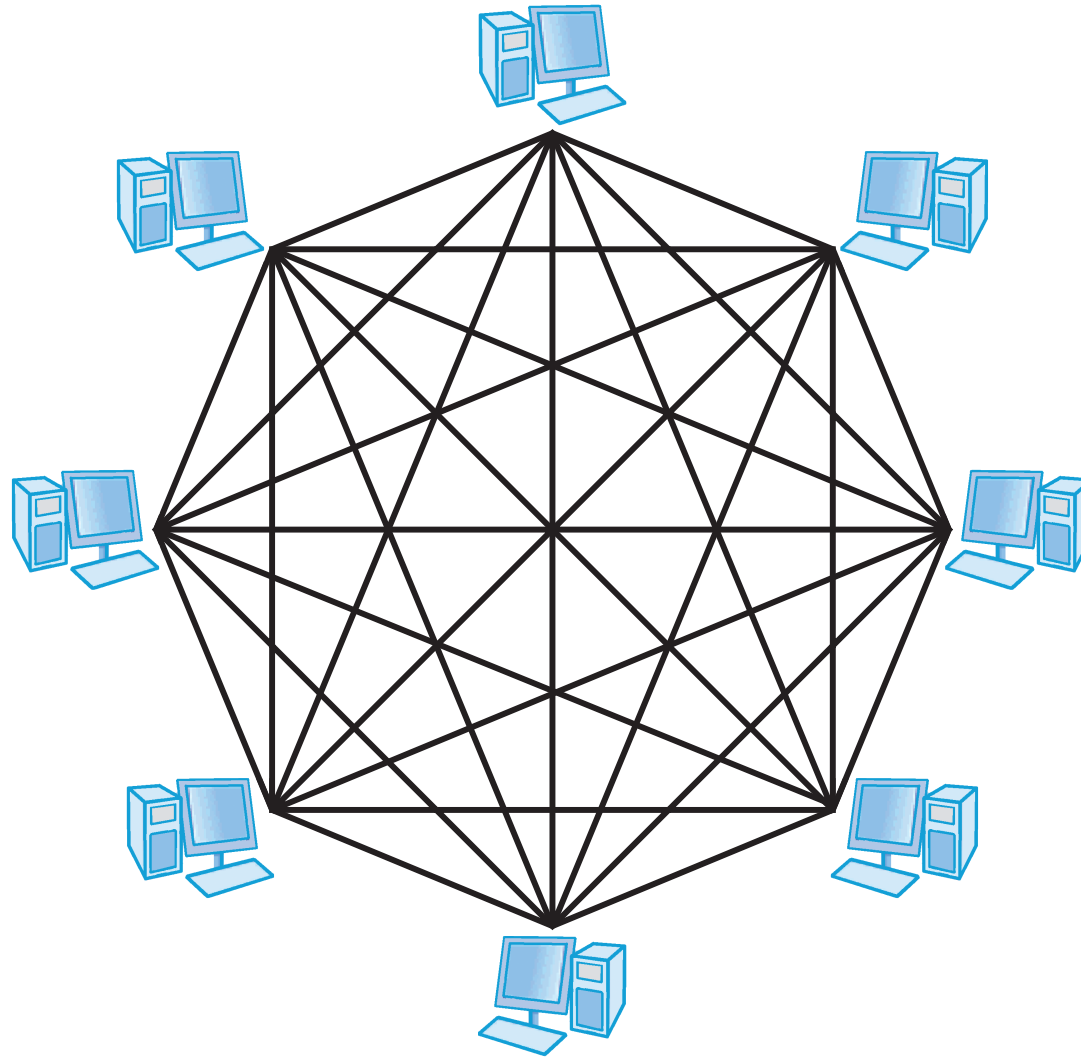
Centralized Perspective



Problems with Centralized Perspective

- Central authority becomes **vulnerable part** of the system:
can be unreliable,
cannot crash,
... ,
- It may not exist (internet applications).

Distributed Perspective



Design Decisions (1)

Two views of the **agent** concept.

- Computer science view:

- **process**,
- collection of interacting agents: a **distributed system**,
- focus on message passing, synchronization, deadlock detection, security, etc.

- Economics point of view:

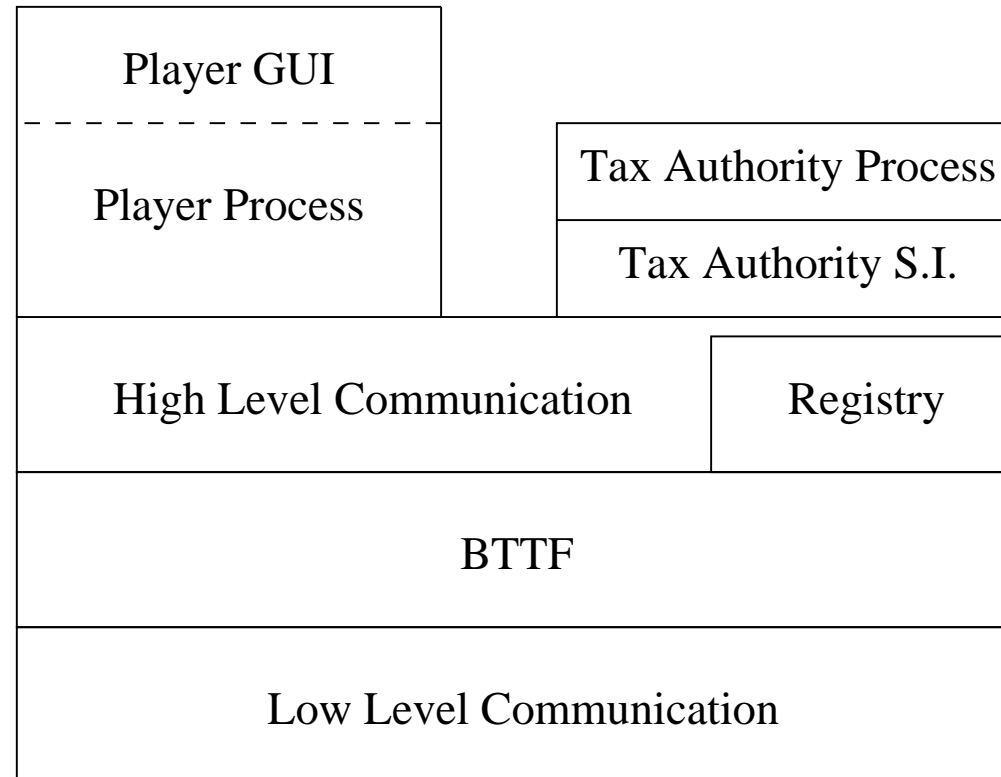
- **player**,
- collection of interacting agents: a **strategic game**,
- focus on utility maximization, private information, truth-telling, equilibrium, etc.

How to reconcile these two views?

Design Decisions (2)

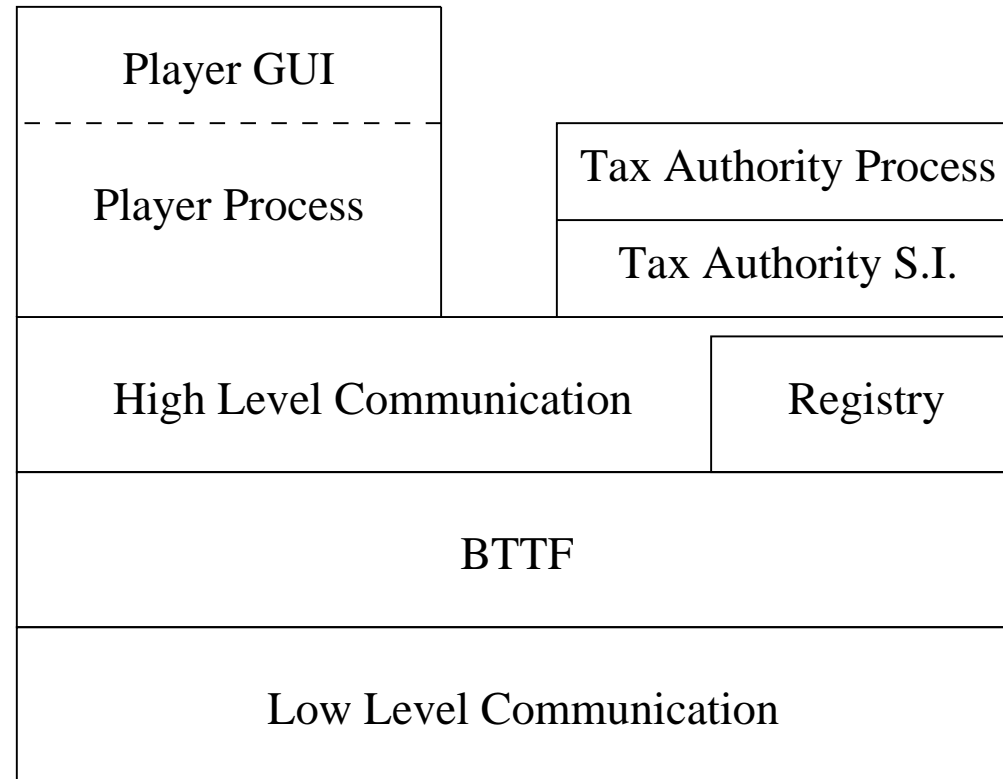
- The system is built as a sequence of **layers**.
- **Lower layers** provide support for **distributed computing**.
- **Upper layers** are concerned only with the matters specific to **mechanism design**.

Architecture (1)



- **Player GUI**: handles interaction with the players (users), so **registration**, **type submission** and **tax reception**.
- Specific mechanisms implemented by instantiating **Player Process**.

Architecture (2)



- **Registered players** pay taxes to each other.
- **Tax authority** collects remaining taxes (if any).

Player Process

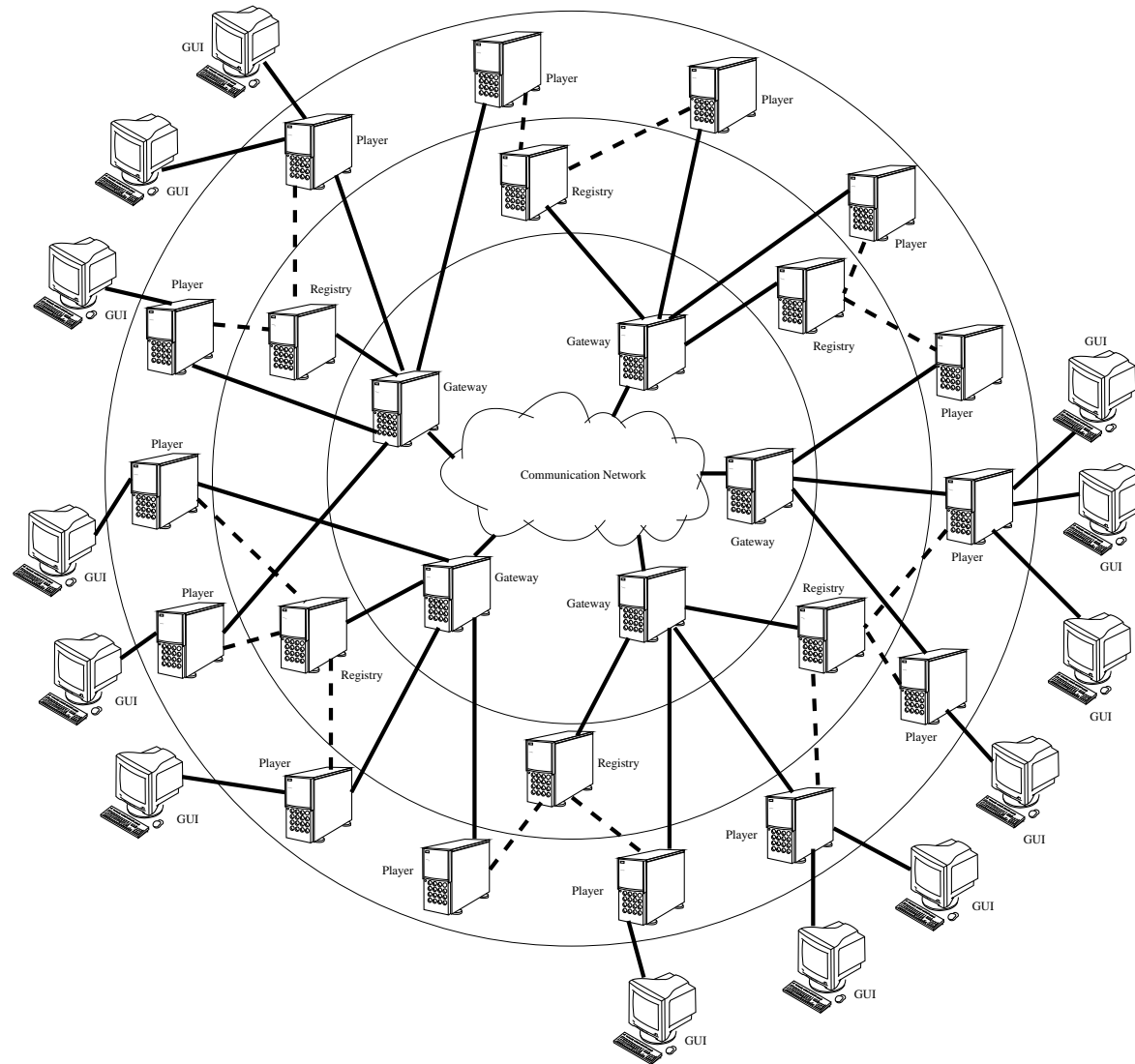
(Simplified Version, Balanced Mechanisms)

- process p_i representing player i is created,
- p_i obtains player i 's type,
- p_i signs in at the local registry,
- all messages sent to p_i are locked and stored,
- if p_i 's registration is confirmed, it broadcasts i 's type (and otherwise it terminates),
- the lock of p_i is open,
- p_i invokes the DTD algorithm. When it ends p_i has received all the types,
- p_i computes decision and tax schemes of the registered players and broadcasts them,
- p_i invokes the DTD algorithm and terminates.

Details

- Lower layers (9K lines of Java code (Kees Blom)),
- Software for message passing between internet-based parallel processes (Han Noot),
- Upper layers (3.5K lines of Java code (Huiye Ma)).

Possible Realization



Implemented Examples

- Vickrey auction (**only 60 lines of code!**),
- Vickrey auction with redistribution (Cavallo '06),
- Public project problems,
- Unit demand auction (uses Kuhn-Munkres algorithm to compute maximum weighted matching),
- Single minded auction (uses a dynamic algorithm developed by V. Markakis),
- Sequential Groves mechanisms (Apt and Estévez-Fernández '07)
- Walker mechanism (Walker '81).

A Demo

Single Minded Auction:

- 5 players,
- 2 local registries,
- tax authority,
- 3 items for sale,
- players bids: A: 20:(1,2), B: 50:(3), C: 32:(2),
D: 60:(2,3), E: 19:(1).
- generated allocation: (3:B, 28), (2:C, 10), (1:E, 0).

The allocation is computed using a dynamic programming algorithm (V. Markakis).

Player A's Interface

Single Minded Auction

register

input your type: x

submit

Information board:

Player B's Interface

Single Minded Auction

register

input your type: x

submit

Information board:

Player A's Interface

Single Minded Auction

input your type:

Information board:

-Phase1: The player has registered successfully.

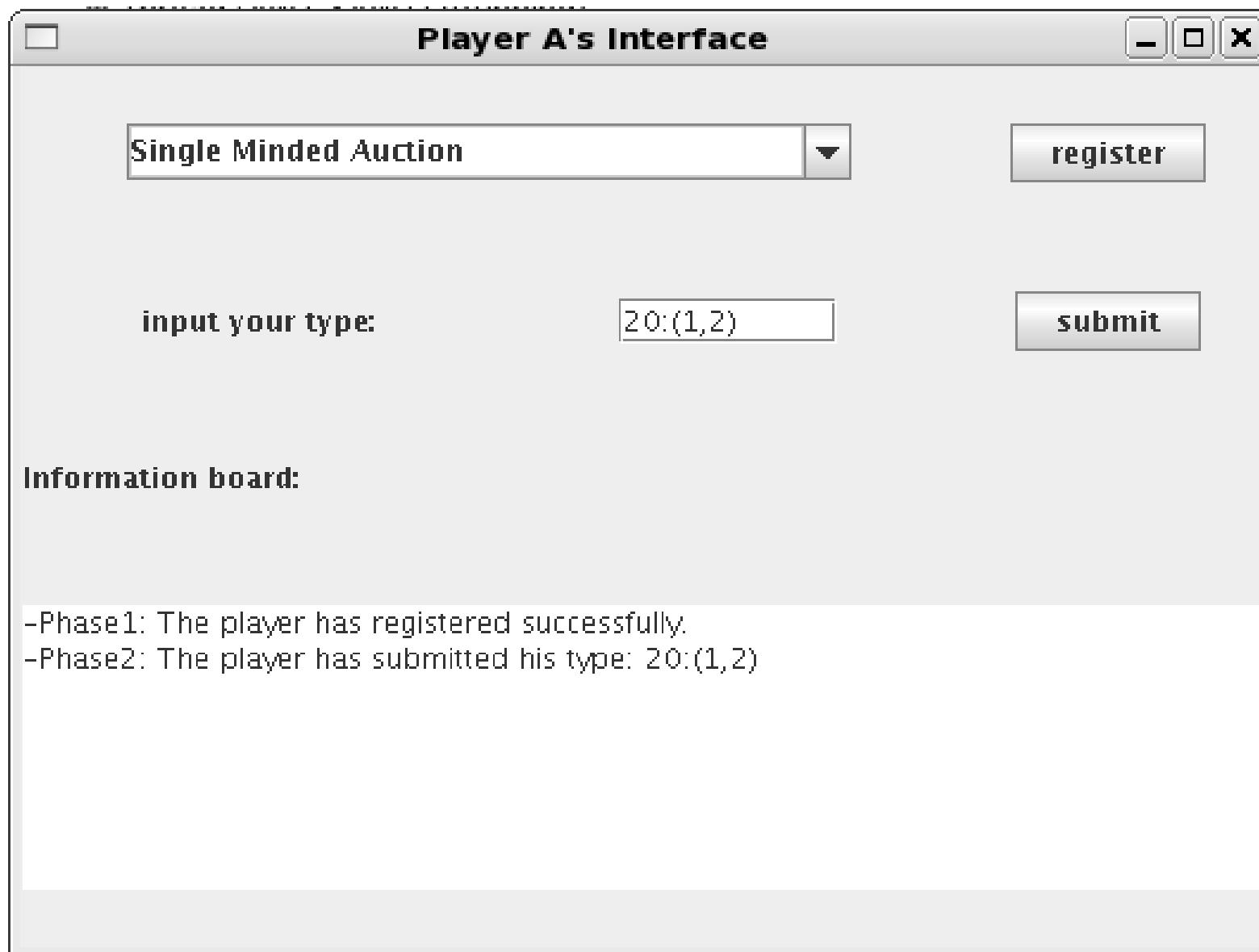
Player B's Interface

Single Minded Auction

input your type:

Information board:

-Phase1: The player has registered successfully.



Player B's Interface

Single Minded Auction

register

input your type: 50:(3)

submit

Information board:

-Phase1: The player has registered successfully.
-Phase2: The player has submitted his type: 50:(3)

Player A's Interface

Single Minded Auction

register

input your type: 20:(1,2)

submit

Information board:

- Phase1: The player has registered successfully.
- Phase2: The player has submitted his type: 20:(1,2)
- Phase3: The player has computed the tax scheme ((3,-1,28),(4,-1,10)) and multicast to others.
- Phase4: The player has received the information about the revenue 38 from the tax authority.
- Phase4: This ends the game.

Player B's Interface

Single Minded Auction

register

input your type: 50:(3)

submit

Information board:

- Phase1: The player has registered successfully.
- Phase2: The player has submitted his type: 50:(3)
- Phase3: The player has received the tax scheme $((3, -1, 28), (4, -1, 10))$ multicast by others.
- Phase3: The player has submitted tax 28 to the tax authority.
- Phase4: The player has received the information about the revenue 38 from the tax authority.
- Phase4: This ends the game.

Player A's Interface

Single Minded Auction

input your type:

Information board:

- Phase1: The player has registered successfully.
- Phase2: The player has submitted his type: 20:(1,2)
- Phase3: The player has computed the tax scheme ((3,-1,28),(4,-1,10)) and multicast to others.
- Phase4: The player has received the information about the revenue 38 from the tax authority.
- Phase4: This ends the game.

Player B's Interface

Single Minded Auction

input your type:

Information board:

- Phase1: The player has registered successfully.
- Phase2: The player has submitted his type: 50:(3)
- Phase3: The player has received the tax scheme ((3,-1,28),(4,-1,10)) multicast by others.
- Phase3: The player has submitted tax 28 to the tax authority.
- Phase4: The player has received the information about the revenue 38 from the tax authority.
- Phase4: This ends the game.

Player C's Interface

Single Minded Auction

input your type:

Information board:

- Phase1: The player has registered successfully.
- Phase2: The player has submitted his type: 32:(2)
- Phase3: The player has computed the tax scheme ((3,-1,28),(4,-1,10)) and multicast to others.
- Phase3: The player has submitted the tax 10 to the tax authority.
- Phase4: The player has received the information about the revenue 38 from the tax authority.
- Phase4: This ends the game.

Player D's Interface

Single Minded Auction

input your type:

Information board:

- Phase1: The player has registered successfully.
- Phase2: The player has submitted his type: 60:(2,3)
- Phase3: The player has received the tax scheme (the computation is finished.) multicast by others.
- Phase4: The player has received the information about the revenue 38 from the tax authority.
- Phase4: This ends the game.

Player E's Interface

Single Minded Auction

input your type:

Information board:

- Phase1: The player has registered successfully.
- Phase2: The player has submitted his type: 19:(1)
- Phase3: The player has computed the tax scheme ((3,-1,28),(4,-1,10)) and multicast to others.
- Phase4: The player has received the information about the revenue 38 from the tax authority.
- Phase4: This ends the game.

Conclusions

- Distributed mechanism design is realistic.
- Our approach
 - is highly flexible,
 - supports fault-tolerance,
 - offers a multi-level protection against manipulations,
 - can be used to implement repeated mechanisms, e.g. continuous auctions.

THANK YOU!