Designing mechanisms for responsive adaptation



Jeff MacKie-Mason
School of Information
and
Department of Economics

Overview



- Develop a principled design methodology
- Benefits
  - replicable
  - extensible
  - applicability expressed explicitly
  - performance quantifiable

Approach: user-centered design

- users are autonomous agents with private information
- system value is aggregation of user values
  - maximize sum of utilities
  - design system to responsively adapt to failures and loss-of-capability
  - include users in design loop: “smart devices, not dumb terminals”

Decentralization requires mechanism design

- Mechanisms:
  - Rules specifying permissible messages
  - Rules specifying allocations as function of messages
- What is a good mechanism? How hard is the problem of mechanism design?

What is mechanism design?

- **What is the design goal?**
  - Efficiency: No feasible allocation can increase system value (weaker form: Pareto efficiency)
- **What requirements constrain design?**
  - Voluntary participation
  - Agent strategies are plausible
  - Computation and complexity costs
- **Design goal cannot always be achieved within constraints**
  - What relaxed constraint sets are feasible?

Example: Applying our methodology to scheduling

- Walsh, Wellman, Wurman, MacKie-Mason (1998)
- Single slot demands:
  - Optima exist
  - Any optimal solution is supported by a price mechanism (but not all price mechanisms are optimal)
- Less is guaranteed for multi-slot demands

We map a spectrum of mechanisms for scheduling

- Single Good <--> Combinatorial <--> GVA
- increasing ---------->
  - scope of concerns
  - cases in which (near) efficient outcome found
  - computational complexity

Example: Bilateral Negotiation

- Anderson, Birgean, MacKie-Mason
- Problem: No mechanism makes all efficient trades while maintaining IR, IC, BB: buyers bid too low, sellers too high
- Approach: Transaction and Entry Fees
  - cover incentive costs, improve efficiency
  - cover actual expenses

Bilateral negotiation cont.

- Mechanisms:
  - GVA: Efficient, not budget balanced
  - k-Double: Budget balanced, not efficient
- Negotiation with fees:
  - transaction
  - entry

Results for bilateral negotiation

- Performance can be improved by multiple-round auction
- Performance depends on nature of management costs (matching principle)

Example: Web caching



- Kelly, Chan, Womer, Jamin, MacKie-Mason
- Caching contributes to quality of network service, survivability
  - cache resources are costly
  - caching needs vary across users, over time of day
  - therefore: cache but exploit user heterogeneity to improve performance

Adaptive, responsive distributed file system hierarchy

- payoffs:
  - faster service ⇳ higher user satisfaction
  - lower requirements for other resources
  - greater system adaptivity: more levers for quality of service
- problems:
  - user values heterogeneous and private
  - free riders

Designing for user heterogeneity

2 clients: Me and Thee

v_{Me}=10, v_{Thee}=1, both make 500 requests

- Suppose Me gets 100 hits; Thee gets 10:
  - $welfare_{Me}$=100*10
  - $welfare_{Thee}$=10*1
  - **Total welfare = 1010**
- With non-market caching, suppose both get 55 hits
  - $welfare_{Me}$= 55*10
  - $welfare_{Thee}$ = 55*1
  - **Total welfare = 605**

Principle: If cache management is determined by heterogenous user valuations, it will perform better than cache that treats all users as identical

Our design approaches



- Market for objects: weighted LFU (Terence)
- Market for space (write-access) (Jonathan and Yee Man)

Recapitulation

- Naïve approach:
  - adaptation requires decentralization,
  - markets are good at solving decentralized allocation problems,
  - so, build markets
- More careful: markets often don't work
  - when and how can we satisfy our desiderata?
  - how well do feasible but non-ideal mechanisms perform?
  - what tradeoffs between the various desiderata?

Method: Model, predict, simulate, implement