CS 194: Distributed Systems

Incentives and Distributed Algorithmic Mechanism Design

Scott Shenker and Ion Stoica
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

1

Traditional Distributed Systems Paradigm

- Choose performance goal
- Design algorithm/protocols to achieve those goals
- Require every node to use that algorithm/protocol

2

Living in the Brave New World....

- Most modern Internet-scale distributed systems involve independent users
  - Web browsing, DNS, etc.
- There is no reason why users have to cooperate
- Users may only care about their own service
- What happens when users behave selfishly?

3

Example: Congestion Control

- Simple model to illustrate basic paradigm
- Users send at rate r_i
- Performance U_i is function of rate and delay
  - use $U_i = r_i/d_i$ for this simple example
- Delay d_i is function of all sending rates r_j
- Selfishness: users adjust their sending rate to maximize their performance

4

Simple Poisson Model with FIFO Queue

- Define $r_{tot} = \Sigma\, r_i$ and $U_{tot} = \Sigma\, U_i$
- In Poisson model with FIFO queues (and link speed 1):

$$d_i = 1/(1-r_{tot})$$

5

Selfish Behavior

- Users adjust r_i to maximize U_i
- We assume they arrive at a Nash equilibrium
- A Nash equilibrium is a vector of r's such that no user can increase their U_i by unilaterally changing r_i
  - First order condition: $\partial U_i/\partial r_i = 0$
- Can be multiple equilibria, or none, but for our example problem there is just one.

6

Nash Equilibrium

- $U_i = r_i(1-r_{tot})$

  $\partial U_i/\partial r_i = 1 - r_{tot} - r_i$
- Solving for all i
- $r_i = 1/(n+1)$ where n is number of users
- $U_{tot} = (n+1)^{-2}$
- Total utility goes down as number of users increases!

7

Socially Optimal Usage

- Set all r_i to be the same value, call it x
- Vary x to maximize U_{tot}

  $U_{tot} = nx(1-nx)$
- Maximizing value is nx = 1/2 and U_{tot} = 1/4 at socially optimal usage
- Huge discrepancy between optimal and selfish outcomes!
  - Why?

8

Fair Queueing

- Very rough model of queueing delays for FQ
- Assume vector of r's is ordered: $r_1 \le r_2 \le r_3 \ldots\ldots \le r_n$
- Smallest flow competes only with own level of usage:

  $d_1 = 1/(1 - nr_1)$
- For all other flows, first r_1 level of packet get this delay also

9

Fair Queueing (continued)

- Packets in $r_2 - r_1$ see delay:

  $1/(1 - r_1 - (n-1)\, r_2)$
- Packets in r3 - r2 see delay:

  $1/(1 - r_1 - r_2 - (n-2)\, r_3)$
- General rule:
  - Everyone gets the same rate at the highest priority (r_1)
  - All remaining flows get the same rate at the next highest priority (r_2)
  - And so on....

10

Nash Equilibrium for FQ

- Nash equilibrium is socially optimal level!
  - Why?
- True for any "reasonable" functions U_i, as long as all users have the same utility
- In general, no users is worse off compared to situation where all users have the same utility as they do

11

Designing for Selfishness

- Assume every user (provider) cares only about their own performance (profit)
- Give each user a set of actions
- Design a "mechanism" that maps action vectors into a system-wide outcome
  - Mechanism design
- Choose a mechanism so that user selfishness leads to socially desirable outcome
  - Nash equilibrium, or other equilibrium concepts

12

Reasons for "Selfish Design" Paradigm

- Necessary to deal with unpleasant reality of selfishness
  - World is going to hell, and the Internet is just going along for the ride.....
- Best way to allow individual users to meet their own needs without enforcing a single "one-size-fits-all" solution
  - With congestion control, everyone must be TCP-compatible
  - That stifles innovation

13

Cooperative vs Noncooperative

- Cooperative paradigm:
  - Works best when all utilities are the same
  - Requires a single standard protocol/algorithm, which inevitably leads to stagnation
  - Is vulnerable to cheaters and malfunctions
- Noncooperative paradigm:
  - Accommodates diversity
  - Allows innovation
  - Does not require enforcement of norms
  - But may not be as efficient....

14

On to a more formal treatment....

- ...combining game theory with more traditional concerns.

15

Three Research Traditions

- Theoretical Computer Science: complexity
  - What can be feasibly computed?
  - Centralized or distributed computational models
- Game Theory: incentives
  - What social goals are compatible with selfishness?
- Internet Architecture: robust scalability
  - How to build large and robust systems?

16

Different Assumptions

- Theoretical Computer Science:
  - Nodes are *obedient*, *faulty*, or *adversarial.*
  - Large systems, limited comp. resources
- Game Theory:
  - Nodes are *strategic* (selfish).
  - Small systems, unlimited comp. resources

17

Internet Systems (1)

- Agents often autonomous (users/ASs)
  - Have their own individual goals
- Often involve "Internet" scales
  - Massive systems
  - Limited comm./comp. resources
- *Both incentives and complexity matter.*

18

Internet Systems (2)

- Agents (users/ASs) are dispersed.
- Computational nodes often dispersed.
- *Computation is (often) distributed.*

19

Internet Systems (3)

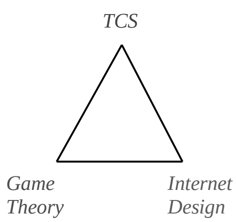
- Scalability and robustness paramount
  - sacrifice strict semantics for scaling
  - many informal design guidelines
  - Ex: end-to-end principle, soft state, *etc.*
- *Computation must be "robustly scalable."*
  - even if criterion not defined precisely
  - *If TCP is the answer, what's the question?*

20

Fundamental Question

What computations are (simultaneously):

- Computationally feasible
- Incentive-compatible
- Robustly scalable



21

Game Theory and the Internet

- Long history of work:
  - Networking: Congestion control [N85], *etc.*
  - TCS: Selfish routing [RT02], *etc.*
- Complexity issues not explicitly addressed
  - though often moot

22

TCS and Internet

- Increasing literature
  - TCP [GY02,GK03]
  - routing [GMP01,GKT03]
  - *etc.*
- No consideration of incentives
- Doesn't always capture Internet style

23

Game Theory and TCS

- Various connections:
  - Complexity classes [CFLS97, CKS81, P85, *etc.*]
  - Cost of anarchy, complexity of equilibria, *etc.* [KP99,CV02,DPS02]
- Algorithmic Mechanism Design (AMD)
  - Centralized computation [NR01]
- Distributed Algorithmic Mechanism Design (DAMD)
  - Internet-based computation [FPS01]

24

DAMD: Two Themes

- Incentives in Internet computation
  - Well-defined formalism
  - Real-world incentives hard to characterize
- Modeling Internet-style computation
  - Real-world examples abound
  - Formalism is lacking

25

System Notation

Outcomes and *agents*:

- Φ is set of possible *outcomes.*
  - $o \in \Phi$ represents particular outcome.
- Agents have valuation functions v_i.
  - $v_i(o)$ is "happiness" with outcome o.

26

Societal vs. Private Goals

- System-wide performance goals:
  - Efficiency, fairness, *etc.*
  - Defined by set of outcomes $G(v) \subset \Phi$
- Private goals: Maximize own welfare
  - v_i is private to agent *i*.
  - Only reveal truthfully if in own interest

27

Mechanism Design

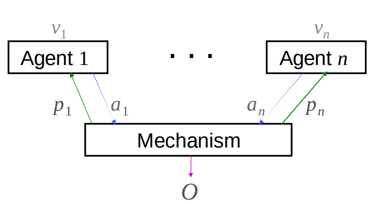
- Branch of game theory:
  - reconciles private interests with social goals
- Involves esoteric game-theoretic issues
  - will avoid them as much as possible
  - only present MD content relevant to DAMD

28

Mechanisms

Actions: a_i Outcome: $O(a)$ Payments: $p_i(a)$

Utilities: $u_i(a) = v_i(O(a)) + p_i(a)$



29

Mechanism Design

- $A_O(v)$ = {action vectors} consistent w/selfishness
  - a_i "maximizes" $u_i(a) = v_i(O(a)) + p_i(a)$.
  - "maximize" depends on information, structure, *etc.*
  - *Solution concept:* Nash, Rationalizable, ESS, *etc.*
- Mechanism-design goal: $O(A_O(v)) \subseteq G(v)$ for all v
- Central MD question: *For given solution concept, which social goals can be achieved?*

30

Direct Strategyproof Mechanisms

- *Direct*: Actions are declarations of v_i.
- *Strategyproof*: $u_i(v) \geq u_i(v_{-i}, x_i)$, for all x_i, v_{-i}
  - Agents have no incentive to lie.
  - $A_O(v) = \{v\}$ *"truthful revelation"*
  - *Example: second price auction*
- *Which social goals achievable with SP?*

31

Strategyproof Efficiency

Efficient outcome: maximizes Σv_i

VCG Mechanism:

- $O(v) = \tilde{o}(v)$ where $\tilde{o}(v) = \arg\max_o \Sigma v_i(o)$
- $p_i(v) = \sum_{j \neq i} v_j(\tilde{o}(v)) + h_i(v_{-i})$

32

Why are VCG Strategyproof?

- Focus only on agent i
  - v_i is truth; x_i is declared valuation
  - $p_i(x_i) = \sum_{j \neq i} v_j(\tilde{o}(x_i)) + h_i$
- $u_i(x_i) = v_i(\tilde{o}(x_i)) + p_i(x_i) = \Sigma_j v_j(\tilde{o}(x_i)) + h_i$
- Recall: $\tilde{o}(v_i)$ maximizes $\Sigma_j v_j(o)$

33

Group Strategyproofness

Definition:

- True: v_i Reported: x_i
- Lying set $S = \{i: v_i \neq x_i\}$

$$\exists\, i \in S\ u_i(x) > u_i(v) \Rightarrow \exists\, j \in S\quad u_j(x) < u_j(v)$$

- *If any liar gains, at least one will suffer.*

34

Algorithmic Mechanism Design [NR01]

Require polynomial-time computability:

- $O(a)$ and $p_i(a)$

Centralized model of computation:

- good for auctions, *etc*.
- not suitable for distributed systems

35

Complexity of Distributed Computations (Static)

Quantities of Interest:

- Computation at nodes
- Communication:
  - total
  - hotspots
- Care about both messages and bits

36

“Good Network Complexity”

- Polynomial-time local computation
  - in total size or (better) node degree
- $O(1)$ messages per link
- Limited message size
  - F(# *agents, graph size, numerical inputs*)

37

Dynamics (partial)

- Internet systems often have “churn.”
  - Agents come and go
  - Agents change their inputs
- “Robust” systems must tolerate churn.
  - most of system oblivious to most changes
- Example of dynamic requirement:
  - $o(n)$ changes trigger $\Omega(n)$ updates.

38

Protocol-Based Computation

- Use standardized protocol as substrate for computation.
  - relative rather than absolute complexity
- Advantages:
  - incorporates informal design guidelines
  - adoption does not require new protocol
  - example: BGP-based mech’s for routing

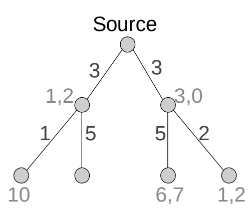
39

Two Examples

- Multicast cost sharing
- Interdomain routing

40

Multicast Cost Sharing (MCS)



Users’ valuations: v_i
Link costs: $c(l)$

Receiver Set
Which users receive the multicast?

Cost Shares
How much does each receiver pay?

Model [FKSS03, §1.2]:
• Obedient Network
• Strategic Users

41

Notation

P	Users (or “participants”)
R	Receiver set ($\sigma_i = 1$ if $i \in R$)
p_i	User i’s cost share (*change in sign!*)
u_i	User i’s utility ($u_i = \sigma_i v_i - p_i$)
W	Total welfare $W(R) = V(R) - C(R)$

$$C(R) = \sum_{l \in T(R)} c(l) \qquad V(R) = \sum_{i \in R} v_i$$

42

“Process” Design Goals

- No Positive Transfers (NPT): $p_i \geq 0$
- Voluntary Participation (VP): $u_i \geq 0$
- Consumer Sovereignty (CS): For all trees and costs, there is a μ_{cs} s.t. $\sigma_i = 1$ if $v_i \geq \mu_{cs}$.
- Symmetry (SYM): If i,j have zero-cost path and $v_i = v_j$, then $\sigma_i = \sigma_j$ and $p_i = p_j$.

43

Two “Performance” Goals

- Efficiency (EFF): $R = \arg\max W$
- Budget Balance (BB): $C(R) = \Sigma_{i \in R}\ p_i$

44

Impossibility Results

Exact [GL79]: No strategyproof mechanism can be both efficient and budget-balanced.

Approximate [FKSS03]: No strategyproof mechanism that satisfies NPT, VP, and CS can be both γ-approximately efficient and κ-approximately budget-balanced, for any positive constants γ, κ.

45

Efficiency

Uniqueness [MS01]: The only strategyproof, efficient mechanism that satisfies NPT, VP, and CS is the Marginal-Cost mechanism (MC):

$$p_i = v_i - (W - W^{-i}),$$

where W is maximal total welfare, and W^{-i} is maximal total welfare without agent i.

- MC also satisfies SYM.

46

Budget Balance (1)

General Construction [MS01]: Any cross-monotonic cost-sharing formula results in a group-strategyproof and budget-balanced cost-sharing mechanism that satisfies NPT, VP, CS, and SYM.

Cost sharing: maps sets to charges $p_i(R)$

Cross-monotonic: shares go down as set increases $p_i(R+j) \leq p_i(R)$

- R is biggest set s. t. $p_i(R) \leq v_i$, for all $i \in R$.

47

Budget Balance (2)

- *Efficiency loss* [MS01]: The Shapley-value mechanism (SH) minimizes the worst-case efficiency loss.
- SH Cost Shares: $c(l)$ is shared equally by all receivers downstream of l.

48

Network Complexity for BB

Hardness [FKSS03]: Implementing a group-strategyproof and budget-balanced mechanism that satisfies NPT, VP, CS, and SYM requires sending $\Omega(|P|)$ bits over $\Omega(|L|)$ links in worst case.

- *Bad network complexity!*

49

Network Complexity of EFF

"*Easiness*" [FPS01]: MC needs only:

- One modest-sized message in each link-direction
- Two simple calculations per node

- *Good network complexity!*

50

Computing Cost Shares

$p_i \equiv v_i - (W - W^{-i})$

Case 1: No difference in tree
Welfare Difference = v_i
Cost Share = 0

Case 2: Tree differs by 1 subtree.
Welfare Difference = W^γ
(minimum welfare subtree above *i*)
Cost Share = $v_i - W^\gamma$

51

Two-Pass Algorithm for MC

Bottom-up pass:

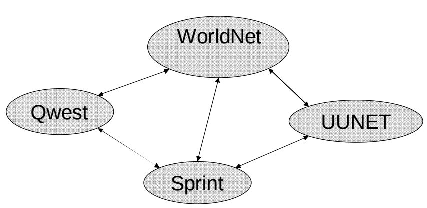
- Compute subtree welfares W^γ.
- If $W^\gamma < 0$, prune subtree.

Top-down pass:

- Keep track of minimum welfare subtrees.
- Compare v_i to minimal W^γ.

52

Interdomain-Routing Mechanism-Design Problem



Agents: Transit ASs
Inputs: Routing Costs or Preferences
Outputs: Routes, Payments

53

Lowest-Cost-Routing MD

Agents' valuations: Per-packet costs $\{c_k\}$

(Unknown) global parameter: Traffic matrix $[T_{ij}]$

Outputs: $\{route(i, j)\}$

Payments: $\{p^k\}$

Objectives:

- Lowest-cost paths (LCPs)
- Strategyproofness
- "BGP-based" distributed algorithm

54

A Unique VCG Mechanism

Theorem [FPSS02]:

For a biconnected network, if LCP routes are always chosen, there is a unique strategyproof mechanism that gives no payment to nodes that carry no transit traffic. The payments are of the form

$$p^k = \sum_{i,j} T_{ij}\, p_{ij}^k, \qquad \text{where}$$

$$p_{ij}^k = c_k + \text{Cost}\,(P^{-k}(c;\, i, j)) - \text{Cost}\,(P(c;\, i, j))$$

Proof is a straightforward application of [GL79].

55

Features of this Mechanism

- Payments have a very simple dependence on traffic $[T_{ij}]$: Payment p^k is weighted sum of per-packet prices p_{ij}^k.
- Cost c_k is independent of i and j, but price p_{ij}^k depends on i and j.
- Price p_{ij}^k is 0 if k is not on LCP between i, j.
- Price p_{ij}^k is determined by cost of min-cost path from i to j not passing through k (min-cost "k-avoiding" path).

56

BGP-Based Computational Model (1)

- Follow abstract BGP model of [GW99]:
  Network is a graph with nodes corresponding to ASs and bidirectional links; intradomain-routing issues are ignored.
- Each AS has a routing table with LCPs to all other nodes:

Dest.	LCP				LCP cost
AS1	AS3	AS5	AS1		3
AS2	AS7	AS2			2

Entire paths are stored, not just next hop.

57

Computational Model (2)

- An AS "advertises" its routes to its neighbors in the AS graph, whenever its routing table changes.
- The computation of a single node is an infinite sequence of stages:

Receive routes from neighbors → Update routing table → Advertise modified routes

- Complexity measures:
  - Number of stages required for convergence
  - Total communication

★ Surprisingly *scalable* in practice.

58

Computing the VCG Mechanism

- Need to compute *routes* and *prices*.
- Routes: Use Bellman-Ford algorithm to compute LCPs and their costs.
- Prices:

$$p_{ij}^k = c_k + \boxed{\text{Cost}\,(P^{-k}(c;\, i, j))} - \text{Cost}\,(P(c;\, i, j))$$

⇒ Need algorithm to compute cost of min-cost k-avoiding path.

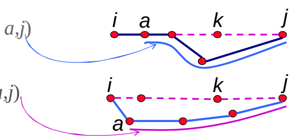
59

Structure of k-avoiding Paths

- BGP uses communication between neighbors only
  ⇒ we need to use "local" structure of $P^{-k}(c;\, i,j)$.
- Tail of $P^{-k}(c;\, i,j)$ is either of the form

(1) $P^{-k}(c;\, a,j)$

or (2) $P(c;\, a,j)$



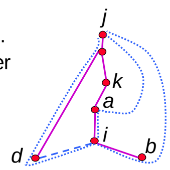
- Conversely, for each neighbor a, either $P^{-k}(c;\, a,j)$ or $P(c;\, a,j)$ gives a candidate for $P^{-k}(c;\, i,j)$.

60

Computing the Prices

- Classifying neighbors:
  - Set of LCPs to j forms a tree.
  - Each of i's neighbors is either
    - (*a*) parent
    - (*b*) child
    - (*d*) unrelated

    in tree of LCPs to j.



- Each case gives a candidate value for p_{ij}^k based on neighbor's LCP cost or price, *e.g.,*

  (b) $p_{ij}^k \leq p_{bj}^k + c_b + c_i$

- p_{ij}^k is the minimum of these candidate values
  ⇒ compute it locally with dynamic programming.

61

A "BGP-Based" Algorithm

Dest.	cost	LCP and path prices				LCP cost
AS1		AS3	AS5	AS1		$c(i,1)$
	c_1	p_{i1}^3	p_{i1}^5			

1. LCPs are computed and advertised to neighbors.
2. Initially, all prices are set to ∞.
3. In the following stages, each node repeats:
   - Receive LCP costs and path prices from neighbors.
   - Recompute candidate prices; select lowest price.
   - Advertise updated prices to neighbors.

Final state: Node i has accurate p_{ij}^k values.

62

Performance of Algorithm

$$d = max_{i,j} \,||\, P(c;\, i, j) \,||$$

$$d' = max_{i,j,k} \,||\, P^{-k}(c;\, i, j) \,||$$

Theorem [FPSS02]:
This algorithm computes the VCG prices correctly, uses routing tables of size $O(nd)$ (a constant factor increase over BGP), and converges in at most $(d + d')$ stages (worst-case additive penalty of d' stages over the BGP convergence time).

63