Distributed Computing Meets Game Theory:
Fault Tolerance and Implementation with Cheap Talk

Joe Halpern
Cornell University

Includes joint work with Ittai Abraham, Danny Dolev, Ivan Geffner, Rica Gonen

Two Views of the World

Work on distributed computing and on cryptography has assumed

- agents are either "good" or "bad"
- good agents follow the protocol
- bad agents do all they can to subvert it

Motivation: the system designer writes a protocol, but some computers might be flaky, and not do what they're supposed to.

Two Views of the World

Work on distributed computing and on cryptography has assumed

- agents are either "good" or "bad"
- good agents follow the protocol
- bad agents do all they can to subvert it

Motivation: the system designer writes a protocol, but some computers might be flaky, and not do what they're supposed to.

Game theory assumes

- all agents are rational
- they try to maximize their utility

Two Views of the World

Work on distributed computing and on cryptography has assumed

- agents are either "good" or "bad"
- good agents follow the protocol
- bad agents do all they can to subvert it

Motivation: the system designer writes a protocol, but some computers might be flaky, and not do what they're supposed to.

Game theory assumes

- all agents are rational
- they try to maximize their utility

Both views make sense in different contexts; we want to combine them.

Thread 1: Fault Tolerance

Byzantine Agreement is a paradigmatic problem in distributed computing:

There are n soldiers; up to t may be faulty.

- n and t are common knowledge

Each soldier starts with an initial preference (1–attack; 0–retreat).
Want an algorithm that (if followed by all the nonfaulty soldiers) guarantees:

- All *nonfaulty* soldiers do the same thing at the same time.
- If all the soldiers are nonfaulty and their initial preferences are identical, that is what they do.

Byzantine Agreement: Results

Typical results:

- With *Byzantine* failures (soldiers can lie and cheat), agreement is possible iff $3t < n$.
- With *crash* failures, agreement is always possible.
- With Byzantine failures and cryptography (messages can be signed with unforgeable signatures), agreement is always possible.
- Agreement (when possible) reachable in $t + 1$ rounds.
- $t + 1$ rounds required, even if no soldiers are actually faulty, all start with the same initial preference, and only crash failures possible.

Byzantine agreement is a game between two teams of unknown composition.

Thread 2: Multiparty Computation

Multiparty computation [Yao '82; Goldreich-Micali-Wigderson '87]: a paradigmatic problem of cryptography.

- Each agent has a secret input.
- Goal: to compute some function of that input, without revealing any information other than the function's output.
  - Just as if a trusted mediator had computed the function

Example: secret input is salary, the function computes highest salary.

There are protocols for multiparty computation, assuming that less than $1/2$ or $1/3$ (depending on underlying assumptions) of the agents are bad.

Mediators

Consider an auction where people do not want to bid publicly

- public bidding reveals useful information
- don't want to do this in bidding for, e.g., oil drilling rights

If there were a mediator (trusted third party), we'd be all set ...

Mediators

Consider an auction where people do not want to bid publicly

- public bidding reveals useful information
- don't want to do this in bidding for, e.g., oil drilling rights

If there were a mediator (trusted third party), we'd be all set ...

- Byzantine agreement can be solved easily with a mediator if $n > 2t$:
  - Each player tells the mediator his preference.
  - The mediator chooses the majority preference.

Thread 3: Implementing Mediators

Implementing mediators is a paradigmatic problem in game theory [Forges 1988/90, Myerson 1986, ...]:

- If a Nash equilibrium (NE) can be achieved with the help of a mediator, can it be achieved using *cheap talk* (i.e., with players just talking to each other)?
- This is almost identical to multiparty communication except:
  - emphasis is on rational players rather than faulty players
  - no concerns about privacy
    - But the solutions provide it

The rest of this talk: combining the threads ...

k-Resilient Equilibria

NE tolerates deviations by one player.

- It's consistent with NE that 2 players could do better by deviating.

An equilibrium is k-*resilient* if no group of size k can gain by deviating (in a coordinated way).

Example: $n > 1$ players must play either 0 or 1.

- if everyone plays 0, everyone gets 1
- if exactly two players play 1, they get 2; the rest get 0.
- otherwise; everyone gets 0.

Everyone playing 0 is a NE, but not 2-resilient.

- Nash equilibrium $=$ 1-resilient equilibrium.
- In general, k-resilient equilibria do not exist if $k > 1$.
- Aumann [1959] already considers resilient equilibria.
- But resilience does not give us all the robustness we need in large systems.

“Irrational” Players

Some agents don’t seem to respond to incentives, perhaps because

- their utilities are not what we thought they were
- they are irrational
- they have faulty computers

Apparently “irrational” behavior is not uncommon:

- People share on Gnutella and Kazaa, seed on BitTorrent

Example:

Consider a group of n bargaining agents.

- If they all stay and bargain, then all get 2.
- Anyone who goes home gets 1.
- Anyone who stays gets 0 if not everyone stays.

Everyone staying is a k-resilient Nash equilibrium for all $k < n$, but not immune to one "irrational" player going home.

- People certainly take such possibilities into account!

Immunity

A protocol is t-*immune* if the payoffs of "good" agents are not affected by the actions of up to t other agents.

- The t agents are like the faulty agents in Byzantine agreement.

A (k, t)-*robust* protocol tolerates coalitions of size k and is t-immune.

- Nash equilibrium = (1,0)-robustness
- In general, (k, t)-robust equilibria don't exist
  - they can be obtained with the help of mediators

Can a (k, t)-robust equilibrium obtained with a mediator be implemented using cheap talk?

Typical Results: Upper Bounds

Theorem 1: Suppose that σ is a (k,t)-robust protocol using a mediator. There is a (k,t)-robust implementation of σ using cheap talk

(a) If $3(k+t) < n$ even if exact utilities are not known;

- protocol runs in *bounded* time

Typical Results: Upper Bounds

Theorem 1: Suppose that σ is a (k, t)-robust protocol using a mediator. There is a (k, t)-robust implementation of σ using cheap talk

(a) If $3(k + t) < n$ even if exact utilities are not known;
  - protocol runs in *bounded* time

(b) If $2k + 3t < n$ and there is a punishment strategy
  - protocol is randomized, has finite *expected* running time

Typical Results: Upper Bounds

Theorem 1: Suppose that σ is a (k, t)-robust protocol using a mediator. There is a (k, t)-robust implementation of σ using cheap talk

(a) If $3(k + t) < n$ even if exact utilities are not known;
  - protocol runs in *bounded* time

(b) If $2k + 3t < n$ and there is a punishment strategy
  - protocol is randomized, has finite *expected* running time

(c) If $2k + 2t < n$ and there is a broadcast channel, with an ϵ error

Typical Results: Upper Bounds

Theorem 1: Suppose that σ is a (k,t)-robust protocol using a mediator. There is a (k,t)-robust implementation of σ using cheap talk

(a) If $3(k+t) < n$ even if exact utilities are not known;
- protocol runs in *bounded* time

(b) If $2k+3t < n$ and there is a punishment strategy
- protocol is randomized, has finite *expected* running time

(c) If $2k+2t < n$ and there is a broadcast channel, with an ϵ error

(d) If $k+t < n$, 1-way functions exist, and there is a punishment strategy, with an ϵ error

The assumptions being made here are all standard assumptions in the distributed computing community.

Typical Results: Upper Bounds

Theorem 1: Suppose that σ is a (k,t)-robust protocol using a mediator. There is a (k,t)-robust implementation of σ using cheap talk

(a) If $3(k+t) < n$ even if exact utilities are not known;
   - protocol runs in *bounded* time

(b) If $2k+3t < n$ and there is a punishment strategy
   - protocol is randomized, has finite *expected* running time

(c) If $2k+2t < n$ and there is a broadcast channel, with an ϵ error

(d) If $k+t < n$, 1-way functions exist, and there is a punishment strategy, with an ϵ error

The assumptions being made here are all standard assumptions in the distributed computing community.

Key idea: reduce to secret sharing + multiparty computation.

Matching Lower Bounds

Theorem 2:

(a) If $3(k+t) \geq n$, $\exists$ a (k,t)-robust strategy using a mediator that cannot be implemented without a mediator without knowing the utilities/without a punishment strategy/in bounded time.

Matching Lower Bounds

Theorem 2:

(a) If $3(k+t) \geq n$, $\exists$ a (k,t)-robust strategy using a mediator that cannot be implemented without a mediator without knowing the utilities/without a punishment strategy/in bounded time.

(b) If $2k+3t \geq n$, $\exists$ a (k,t)-robust strategy with a mediator that cannot be simulated without a mediator, even if there is a punishment strategy and utilities are known.

Matching Lower Bounds

Theorem 2:

(a) If $3(k+t) \geq n$, $\exists$ a (k,t)-robust strategy using a mediator that cannot be implemented without a mediator without knowing the utilities/without a punishment strategy/in bounded time.

(b) If $2k+3t \geq n$, $\exists$ a (k,t)-robust strategy with a mediator that cannot be simulated without a mediator, even if there is a punishment strategy and utilities are known.

(c) If $2k+2t \geq n \ldots$

Matching Lower Bounds

Theorem 2:

(a) If $3(k+t) \geq n$, $\exists$ a (k,t)-robust strategy using a mediator that cannot be implemented without a mediator without knowing the utilities/without a punishment strategy/in bounded time.

(b) If $2k+3t \geq n$, $\exists$ a (k,t)-robust strategy with a mediator that cannot be simulated without a mediator, even if there is a punishment strategy and utilities are known.

(c) If $2k+2t \geq n \ldots$

(d) $k+t \geq n \ldots$

Some proofs exploit techniques used in lower bound proofs for Byzantine agreement.

Lower Bounds on Running Time

Theorem 3: If $2k + 2t \geq n$, then

(a) there is a game Γ with a (k, t)-robust strategy with a mediator that cannot be implemented by *any* deterministic cheap talk strategy.

Lower Bounds on Running Time

Theorem 3: If $2k + 2t \geq n$, then

(a) there is a game Γ with a (k, t)-robust strategy with a mediator that cannot be implemented by *any* deterministic cheap talk strategy.

(b) for all b, there is a game Γ_b with a (k, t)-robust strategy with a mediator that cannot be implemented using cheap talk with expected running time $\leq b$.

Lower Bounds on Running Time

Theorem 3: If $2k + 2t \geq n$, then

(a) there is a game Γ with a (k,t)-robust strategy with a mediator that cannot be implemented by *any* deterministic cheap talk strategy.

(b) for all b, there is a game Γ_b with a (k,t)-robust strategy with a mediator that cannot be implemented using cheap talk with expected running time $\leq b$.

(c) there is a game Γ with a (k,t)-robust strategy with a mediator such that for all ϵ, there exists b_ϵ such that we cannot implement the mediator with ϵ error with a cheap-talk strategy that runs in $\leq b_\epsilon$ steps.

Asynchronous Systems

All these results assume that systems are synchronous.

- Players communicate with each other, and then all make a decision in the same round.
- But why should end of cheap talk be common knowledge?
- Asynchrony is a common feature is many real-world applications
  - Markets are asynchronous!
  - Blockchain assumes partial synchrony

Implementation in Asynchronous Systems

What does it mean to implement a mediator in asynchronous systems?

- Issue: the outcome might depend on the scheduler
  - What order players are scheduled in
  - How long messages take to arrive

We want it to be the case that, for each scheduler in the mediator game, there is a scheduler that implements the same outcome in the communication game, and vice versa.

Implementation in Asynchronous Systems

What does it mean to implement a mediator in asynchronous systems?

- Issue: the outcome might depend on the scheduler
  - What order players are scheduled in
  - How long messages take to arrive

We want it to be the case that, for each scheduler in the mediator game, there is a scheduler that implements the same outcome in the communication game, and vice versa.

Theorem 4: Suppose that σ is a (k,t)-robust protocol using a mediator. There is a (k,t)-robust implementation of σ using cheap talk

(a) If $4(k+t) < n$ even if exact utilities are not known;

(b) If $3k + 4t < n$ and there is a punishment strategy.

Related Work

Lots of related work on implementation in both CS and game theory:

- work of Forges + Barany [≈1990] gives Theorem 1(a) with $k = 1$
- work on secure multiparty [BGW88,CCD88] computation gives Theorem 1(a) for all (k, t)!

Related Work

Lots of related work on implementation in both CS and game theory:

- work of Forges + Barany [≈1990] gives Theorem 1(a) with $k = 1$
- work on secure multiparty [BGW88,CCD88] computation gives Theorem 1(a) for all (k, t)!
- Ben-Porath ('03): Theorem 1(b) with $k = 1$ (no crypto, known utilities, but does sequential equilibrium)
- Heller ('05): extends B-P to all k; proves matching lower bound
- Theorem 3(a) shows that B-P's strategy is incorrect (because bounded); Heller's has problems too
  - B-P has a correction using *verifiability*, an unimplementable assumption

More Related Work

- Lysanskaya-Triandopoulos: Theorem 1(c) for $k = 1$
- Rabin/Ben-Or's work essentially gives Theorem 1(c) for all (k, t)
- Urbano-Vila ('04) and Dodis-Halevi-Rabin ('00) get Theorem 1(d) if $k = 1$, $n = 2$
- Theorem 3(a) shows UV's strategy is incorrect
- Izmalkov, Micali, Lepinski; Lepinski, Micali, Shelat ('05) prove stronger implementation results, but require strong primitives (*envelopes* and *ballot-boxes*) that cannot be implemented over broadcast channels

Conclusions

- Issues of coalitions and fault-tolerance are critical in distributed computing, game theory, and cryptography.
- By combining ideas from all three areas we can gain new insights, and prove interesting new results.

Conclusions

- Issues of coalitions and fault-tolerance are critical in distributed computing, game theory, and cryptography.
- By combining ideas from all three areas we can gain new insights, and prove interesting new results.

Some implications for distributed computing/cryptography:

- We should consider rational players as well as Byzantine players
  - This could lead to new protocols (indeed, it already has)

Conclusions

- Issues of coalitions and fault-tolerance are critical in distributed computing, game theory, and cryptography.
- By combining ideas from all three areas we can gain new insights, and prove interesting new results.

Some implications for distributed computing/cryptography:

- We should consider rational players as well as Byzantine players
  - This could lead to new protocols (indeed, it already has)
- We may also want to consider obedient/altruistic players
  - In real life, people are often willing to follow instructions, provided they don't get too badly hurt
  - Protocol/mechanism designers should take advantage of that!

Implications for Game Theory

- Equilibrium notions should be more robust, and take fault tolerance into account
- Cryptographic techniques can be helpful in achieving equilibrium

Other ideas from distributed computing/crypography may be relevant:

- Resource-bounded equilibria
- Synchrony vs. asynchrony