Optimal Eventual Byzantine Agreement Protocols with Omission Failures

PODC 2023

Kaya Alpturer[1] Joseph Y. Halpern[1] Ron van der Meyden[2]

[1]Cornell University, USA

[2]UNSW Sydney, Australia

Eventual Byzantine Agreement (EBA)

There are n agents $Agt = \{1, \ldots, n\}$, up to t of which are faulty.

Agents have initial preferences for which value (0 or 1) to decide on. They each can perform an action $\texttt{decide}_i(v)$.

Eventual Byzantine Agreement (EBA)

There are n agents $Agt = \{1, \ldots, n\}$, up to t of which are faulty.

Agents have initial preferences for which value (0 or 1) to decide on. They each can perform an action $\texttt{decide}_i(v)$.

We require, for each value $v \in \{0, 1\}$,

- **Unique Decision:** all agents decide at most once
- **Agreement:** nonfaulty agents decide on the same value
- **Validity:** if a nonfaulty agent decides v, then some agent had initial preference v
- **Termination:** nonfaulty agents eventually decide

Communication Model

We focus on the *round-based, synchronous, message-passing* communication model with omission failures.

- **Sending-omission Failures:** A faulty agent may omit to send an arbitrary set of messages in any given round.

Designing Optimal Protocols

Dwork and Moses [1990][1] defined a notion of optimality for Byzantine agreement protocols.

P_1 **dominates protocol** P_2**:** if for all *corresponding* runs r_1 and r_2, for all agents j, we have $dt_1(r_1) \leq dt_2(r_2)$, where $dt_i(r_i)$ is the decision time of agent j running P_i in run r_i.

- A run r_1 of P_1 *corresponds* to a run r_2 of P_2 if r_1 and r_2 agree on all agents' inputs and the *failure pattern*
  - which agents are faulty and which of their messsages are sent

[1] C. Dwork and Y. Moses. 1990. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation 88*, 2 (1990), 156–186.

Designing Optimal Protocols

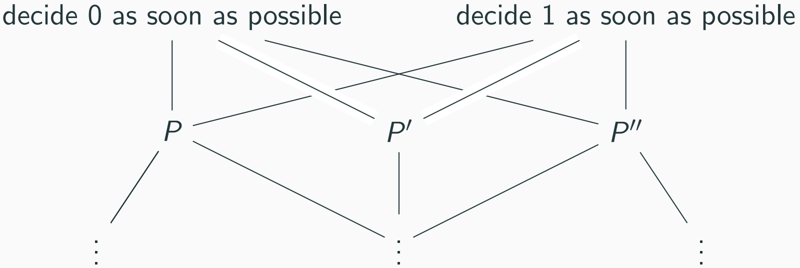
Optimal protocols:

- For EBA, *optimal* protocols are the ones that are not dominated by any other protocol.
  - E.g., there is an optimal protocol where (roughly speaking) agents decide 0 as soon as possible, and one where agents decide 1 as soon as possible
    - neither dominates the other

Designing Optimal Protocols

Optimal protocols:

- For EBA, *optimal* protocols are the ones that are not dominated by any other protocol.
  - E.g., there is an optimal protocol where (roughly speaking) agents decide 0 as soon as possible, and one where agents decide 1 as soon as possible
    - neither dominates the other



Working with Limited Information

How much information should the agents send with each message if we want to design an optimal protocol?

Working with Limited Information

How much information should the agents send with each message if we want to design an optimal protocol?

Typical methodology: Without loss of generality, use a full-information protocol, then find the best decision rule.

Working with Limited Information

How much information should the agents send with each message if we want to design an optimal protocol?

Typical methodology: Without loss of generality, use a full-information protocol, then find the best decision rule.

But full information is costly, especially with limited bandwidth.

- What happens if we limit information exchange?

Our Main Idea

We can describe a protocol in terms of two separate components:

- the *information-exchange* protocol
    - determines what information is exchanged
- the *action* protocol,
    - determines when to decide on a value.

Our Main Idea

We can describe a protocol in terms of two separate components:

- the *information-exchange* protocol
  - determines what information is exchanged
- the *action* protocol,
  - determines when to decide on a value.

BA Protocol

action protocol P information-exchange protocol $\mathcal{E}$

A protocol P is optimal w.r.t. an information-exchange protocol $\mathcal{E}$ if P is optimal amongst protocols that use $\mathcal{E}$.

Our Results

- We provide a knowledge-based program $\mathbf{P}^0$ that gives an optimal protocol in two limited-information settings.

[2] J. Y. Halpern, Y. Moses, and O. Waarts. 2001. A characterization of eventual Byzantine agreement. SIAM J. Comput. 31, 3 (2001), 838–865. https://doi.org/10.1137/S0097539798340217

Our Results

- We provide a knowledge-based program $\mathbf{P}^0$ that gives an optimal protocol in two limited-information settings.
- We a provide a knowledge-based program $\mathbf{P}^1$ that generalizes $\mathbf{P}^0$ and gives an optimal protocol w.r.t. full-information exchange.

[2] J. Y. Halpern, Y. Moses, and O. Waarts. 2001. A characterization of eventual Byzantine agreement. SIAM J. Comput. 31, 3 (2001), 838–865. https://doi.org/10.1137/S0097539798340217

Our Results

- We provide a knowledge-based program $\mathbf{P}^0$ that gives an optimal protocol in two limited-information settings.
- We a provide a knowledge-based program $\mathbf{P}^1$ that generalizes $\mathbf{P}^0$ and gives an optimal protocol w.r.t. full-information exchange.
- Moreover, $\mathbf{P}^1$ is implementable in polynomial time, which shows that a polynomial-time optimal protocol for EBA with omission failures exists, settling a question left open by Halpern, Moses, and Waarts[2] (HMW from now on).

[2] J. Y. Halpern, Y. Moses, and O. Waarts. 2001. A characterization of eventual Byzantine agreement. SIAM J. Comput. 31, 3 (2001), 838–865. https://doi.org/10.1137/S0097539798340217

Reasoning about Knowledge

To model these systems semantically, we use the standard runs-and-systems model [Fagin et al., 1995].[3]

[3] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. 1995. *Reasoning About Knowledge*. MIT Press, Cambridge, MA. A slightly revised paperback version was published in 2003.

Reasoning about Knowledge

To model these systems semantically, we use the standard runs-and-systems model [Fagin et al., 1995].[3]

An *interpreted system* is a pair $\mathcal{I} = (\mathcal{R}, \pi)$.

- $\mathcal{R}$ is the set of runs.
- π is an *interpretation function* that indicates which atomic propositions are true at each point (r, m) in the system.

[3] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. 1995. *Reasoning About Knowledge*. MIT Press, Cambridge, MA. A slightly revised paperback version was published in 2003.

Reasoning about Knowledge

To model these systems semantically, we use the standard runs-and-systems model [Fagin et al., 1995].[3]

An *interpreted system* is a pair $\mathcal{I} = (\mathcal{R}, \pi)$.

- $\mathcal{R}$ is the set of runs.
- π is an *interpretation function* that indicates which atomic propositions are true at each point (r, m) in the system.

Formally, a run $r \in \mathcal{R}$ is a function mapping a time m to a *global state* $r(m)$:

- a tuple $(s_e, s_1, \ldots, s_n)$ describing the local state of the environment and the local state of each agent $i \in \{1, \ldots, n\}$.

[3] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. 1995. *Reasoning About Knowledge*. MIT Press, Cambridge, MA. A slightly revised paperback version was published in 2003.

Reasoning about Knowledge

The semantics of the logic is given by $\mathcal{I}, (r, m) \models \phi$ where ϕ is a formula. Key definition:

- $(\mathcal{I}, (r, m)) \models K_i\phi$ if $(\mathcal{I}, (r', m)) \models \phi$ at all points (r', m) such that i has the same local state in (r, m) and (r', m)

Another useful formula for EBA:

- $(\mathcal{I}, (r, m)) \models \exists 0$ if some agent's initial state in r is 0.

Knowledge-based programs

Knowledge-based programs [Halpern and Fagin, 1985][4] are high-level abstractions.

[4] J. Y. Halpern and R. Fagin. 1985. A formal model of knowledge, action, and communication in distributed systems: preliminary report. *In Proc. 4th ACM Symposium on Principles of Distributed Computing*. 224–236.

Knowledge-based programs

Knowledge-based programs [Halpern and Fagin, 1985][4] are high-level abstractions.

- *Standard programs* are ones where the tests directly depend on the agent i's local state.
- For knowledge-based programs, the tests can be Boolean combinations of $K_i\psi$

$$\text{if } K_i(\exists 0) \text{ then } \texttt{decide}_i(0) \textit{ else } \ldots$$

A standard program P *implements* a knowledge-based program $\mathbf{P}$ in an intepreted system $\mathcal{I}$ if the knowledge-based conditions in $\mathbf{P}$ hold at points in $\mathcal{I}$ exactly when the standard conditions in P hold.

$$\text{if } \mathit{init}_i = 0 \wedge \text{received}_i(0) \text{ then } \texttt{decide}_i(0) \textit{ else } \ldots$$

[4] J. Y. Halpern and R. Fagin. 1985. A formal model of knowledge, action, and communication in distributed systems: preliminary report. *In Proc. 4th ACM Symposium on Principles of Distributed Computing*. 224–236.

Crash $\rightarrow$ omission failures

One idea is to decide 0 agressively [HMW, Castañeda et al., 2014][5]

[5] A. Castañeda, Y. A. Gonczorowski, and Y. Moses. 2014. Unbeatable consensus. *In Proc. 28th International Conference on Distributed Computing (DISC '14)*. 91–106.

Crash $\rightarrow$ omission failures

One idea is to decide 0 agressively [HMW, Castañeda et al., 2014][5]

Consider a protocol that decides 0 when $K_i(\exists 0)$ holds.

[5] A. Castañeda, Y. A. Gonczorowski, and Y. Moses. 2014. Unbeatable consensus. *In Proc. 28th International Conference on Distributed Computing (DISC '14)*. 91–106.

Crash $\rightarrow$ omission failures

One idea is to decide 0 agressively [HMW, Castañeda et al., 2014][5]

Consider a protocol that decides 0 when $K_i(\exists 0)$ holds.

- This works for crash failures, but with omission failures we run into a problem.
- Agent i can't always decide 0 when $K_i(\exists 0)$ holds when we have omission failures.

[5] A. Castañeda, Y. A. Gonczorowski, and Y. Moses. 2014. Unbeatable consensus. *In Proc. 28th International Conference on Distributed Computing (DISC '14)*. 91–106.

Knowledge-based program $\mathbf{P}^0$

The solution is to decide 0 when we know someone has *just decided* 0 in the previous round.

Knowledge-based program $\mathbf{P}^0$

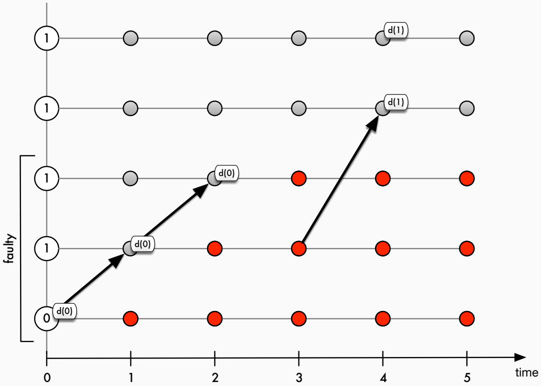
The solution is to decide 0 when we know someone has *just decided* 0 in the previous round.

- This means we decide 0 when there is a chain of 0-decisions, going back to the first round

Knowledge-based program P^0

The solution is to decide 0 when we know someone has *just decided* 0 in the previous round.

- This means we decide 0 when there is a chain of 0-decisions, going back to the first round



Knowledge-based program P^0

- Agent *i* decides 0 if *i* knows that someone *just decided* 0 or *i* has initial preference 0.
- Agent *i* decides 1 if *i* knows that no agent is currently deciding 0.

Knowledge-based program $\mathbf{P}^0$

- Agent i decides 0 if i knows that someone *just decided* 0 or i has initial preference 0.
- Agent i decides 1 if i knows that no agent is currently deciding 0.

Program: $\mathbf{P}_i^0$

if $\mathit{decided}_i \neq \bot$ **then** `noop`
else if $\mathit{init}_i = 0 \ \vee \ K_i(\bigvee_{j \in Agt} \mathit{jdecided}_j = 0)$ **then** $\texttt{decide}_i(0)$
else if $K_i(\bigwedge_{j \in Agt} \neg(\mathit{deciding}_j = 0))$ **then** $\texttt{decide}_i(1)$
else `noop`

This represents the *action protocol*. We will investigate it with respect to different information-exchange protocols.

Optimality of $\mathbf{P}^0$

In the paper, we give a sufficient condition for the optimality of $\mathbf{P}^0$ with respect to different information exchanges.

Optimality of $\mathbf{P}^0$

In the paper, we give a sufficient condition for the optimality of $\mathbf{P}^0$ with respect to different information exchanges.

We then define two limited-information settings, $\mathcal{E}_{\min}$ and $\mathcal{E}_{basic}$, that satisfy this condition.

P^0 is optimal in two limited-information settings

In $\mathcal{E}_{\mathrm{min}}$, agents keep track only of the time, their initial value, whether they have decided, and whether they received a message from another agent.

$\mathbf{P}^0$ is optimal in two limited-information settings

In $\mathcal{E}_{\mathrm{min}}$, agents keep track only of the time, their initial value, whether they have decided, and whether they received a message from another agent.

- **Local states:**
  - $time_i$: records the time
  - $init_i$: records the initial preference of the agent
  - $decided_i$: records the agent's decision
  - jd_i: records whether agent i received a message from an agent that decided.

$\mathbf{P}^0$ is optimal in two limited-information settings

In $\mathcal{E}_{\mathrm{min}}$, agents keep track only of the time, their initial value, whether they have decided, and whether they received a message from another agent.

- **Local states:**
  - $time_i$: records the time
  - $init_i$: records the initial preference of the agent
  - $decided_i$: records the agent's decision
  - jd_i: records whether agent i received a message from an agent that decided.
- **Messages:** 0 or 1. Each agent send a message once, only when they decide.

Now we consider the action protocol that implements $\mathbf{P}^0$ with respect to this information exchange.

Implementing $\mathbf{P}^0$ in $\mathcal{E}_{\mathsf{min}}$

The following standard program implements $\mathbf{P}^0$ in $\mathcal{E}_{\mathsf{min}}$:

- agent i decides 0 if it has an initial value of 0 or it received the message 0 in the previous round.
- agent i decides 1 in round $t+1$ if it hasn't decided 0

Implementing $\mathbf{P}^0$ in $\mathcal{E}_{\mathrm{min}}$

The following standard program implements $\mathbf{P}^0$ in $\mathcal{E}_{\mathrm{min}}$:

- agent i decides 0 if it has an initial value of 0 or it received the message 0 in the previous round.
- agent i decides 1 in round $t+1$ if it hasn't decided 0

Program: P_i^{min}

if $decided_i \neq \bot$ **then** `noop`
else if $init_i = 0 \vee jd_i = 0$ **then** $\texttt{decide}_i(0)$
else if $time_i = t+1$ **then** $\texttt{decide}_i(1)$
else `noop`

This is just the usual EBA program; it is optimal with limited-information exchange!

P^0 is optimal in two limited-information settings

In $\mathcal{E}_{basic}$, agents keep track of everything in $\mathcal{E}_{\min}$ + how many messages of the form $(init, 1)$ (which encodes "I haven't heard about a 0") they received in the last round.

$\mathbf{P}^0$ is optimal in two limited-information settings

In $\mathcal{E}_{basic}$, agents keep track of everything in $\mathcal{E}_{\min}$ + how many messages of the form $(init, 1)$ (which encodes "I haven't heard about a 0") they received in the last round.

- **Local states:** same as in $\mathcal{E}_{min}$ except one additional variable,
  - $\#1_i$ counts the number of messages of the form $(init, 1)$ that i receives in the previous round.
- **Messages:** 0, 1 or $(init, 1)$.
  - 0 (resp., 1) is sent when the agent decides 0 (resp., 1);
  - $(init, 1)$ is sent if the agent hasn't decided yet.

Now, the action protocol that implements $\mathbf{P}^0$ can do better compared to the action protocol that implements $\mathbf{P}^0$ in the minimal information-exchange.

Implementing $\mathbf{P}^0$ in $\mathcal{E}_{basic}$

The following standard program implements $\mathbf{P}^0$ in $\mathcal{E}_{basic}$:

Now agent i can decide if it hears from enough agents that they haven't decided 0.

- This is guaranteed to happen by round $t + 1$, but may happen earlier

Implementing $\mathbf{P}^0$ in $\mathcal{E}_{basic}$

The following standard program implements $\mathbf{P}^0$ in $\mathcal{E}_{basic}$:

Now agent i can decide if it hears from enough agents that they haven't decided 0.

- This is guaranteed to happen by round $t + 1$, but may happen earlier

Program: P_i^{basic}

if $\mathit{decided}_i \neq \bot$ **then** `noop`
else if $\mathit{init}_i = 0 \vee \mathit{jd}_i = 0$ **then** $\texttt{decide}_i(0)$
else if $\#1_i > n - \mathit{time}_i \vee \mathit{jd}_i = 1$ **then** $\texttt{decide}_i(1)$
else `noop`

Is P^0 optimal w.r.t. full-information exchange?

No.

Is P^0 optimal w.r.t. full-information exchange?

No.

Example: $n = 20$, $t = 10$. Every agent has initial preference 1, and no faulty agent ever sends a message.

Is P^0 optimal w.r.t. full-information exchange?

No.

Example: $n = 20$, $t = 10$. Every agent has initial preference 1, and no faulty agent ever sends a message.

By the end of the first round, all nonfaulty agents know who the faulty agents are.

Is P^0 optimal w.r.t. full-information exchange?

No.

Example: $n = 20$, $t = 10$. Every agent has initial preference 1, and no faulty agent ever sends a message.

By the end of the first round, all nonfaulty agents know who the faulty agents are.

By the end of the second round, it is common knowledge among the nonfaulty agents who the faulty agents are, and that the nonfaulty agents all started with 1.

Is P^0 optimal w.r.t. full-information exchange?

No.

Example: $n = 20$, $t = 10$. Every agent has initial preference 1, and no faulty agent ever sends a message.

By the end of the first round, all nonfaulty agents know who the faulty agents are.

By the end of the second round, it is common knowledge among the nonfaulty agents who the faulty agents are, and that the nonfaulty agents all started with 1.

- It can be shown that this suffices for the nonfaulty agents to decide 1 in round 3.
- However, with $\mathbf{P}^0$, they don't decide until round 11
  - no agent knows that there is no chain of 0s until round 11.

Common knowledge

We can get an optimal protocol in the full-information setting by taking advantage of common knowledge.

Common knowledge

We can get an optimal protocol in the full-information setting by taking advantage of common knowledge.

- Every nonfaulty agent knows φ: $E_{\mathcal{N}}\varphi$
- φ is common knowledge among the nonfaulty agents: $C_{\mathcal{N}}\varphi$
    - All the nonfaulty agents know φ, all the nonfaulty agents know that all the nonfaulty agents know φ, ...

$$C_{\mathcal{N}}\varphi \quad \Leftrightarrow \quad E_{\mathcal{N}}\varphi \wedge E_{\mathcal{N}}E_{\mathcal{N}}\varphi \wedge E_{\mathcal{N}}E_{\mathcal{N}}E_{\mathcal{N}}\varphi \wedge \cdots$$

Knowledge-based program $\mathbf{P}^1$

We show that a necessary condition for optimality w.r.t. full-information exchange is:

Knowledge-based program $\mathbf{P}^1$

We show that a necessary condition for optimality w.r.t. full-information exchange is:

- if an undecided agent i knows that it is common knowledge among the nonfaulty agents who the faulty agents are ($C_{\mathcal{N}}\,t\text{-}\mathit{faulty}$) then it must make a decision.

Knowledge-based program $\mathbf{P}^1$

We show that a necessary condition for optimality w.r.t. full-information exchange is:

- if an undecided agent i knows that it is common knowledge among the nonfaulty agents who the faulty agents are ($C_{\mathcal{N}}\, \textit{t-faulty}$) then it must make a decision.

We can modify $\mathbf{P}^0$ to get $\mathbf{P}^1$, which makes use of this:

Program: $\mathbf{P}_i^1$

if $\textit{decided}_i \neq \bot$ **then** `noop`
else if $K_i(C_{\mathcal{N}}(\textit{t-faulty} \wedge \textit{no-decided}_{\mathcal{N}}(1) \wedge \exists 0)$ **then** $\texttt{decide}_i(0)$
else if $K_i(C_{\mathcal{N}}(\textit{t-faulty} \wedge \textit{no-decided}_{\mathcal{N}}(0) \wedge \exists 1))$ **then** $\texttt{decide}_i(1)$
else if $\textit{init}_i = 0 \ \vee\ K_i(\bigvee_{j \in Agt} \textit{jdecided}_j = 0)$ **then** $\texttt{decide}_i(0)$
else if $K_i(\bigwedge_{j \in Agt} \neg(\textit{deciding}_j = 0))$ **then** $\texttt{decide}_i(1)$
else `noop`

Showing P^1 is optimal

In the paper, we show that

Showing P^1 is optimal

In the paper, we show that

- $\mathbf{P}^1$ is equivalent to $\mathbf{P}^0$ in the limited-exchange settings we considered earlier

Showing $\mathbf{P}^1$ is optimal

In the paper, we show that

- $\mathbf{P}^1$ is equivalent to $\mathbf{P}^0$ in the limited-exchange settings we considered earlier
- $\mathbf{P}^1$ is also optimal in the full-information exchange setting.

Showing $\mathbf{P}^1$ is optimal

In the paper, we show that

- $\mathbf{P}^1$ is equivalent to $\mathbf{P}^0$ in the limited-exchange settings we considered earlier
- $\mathbf{P}^1$ is also optimal in the full-information exchange setting.
    - The proof makes use of the *continual common knowledge* operator $C_{\mathcal{S}}^{\boxdot}\phi$ introduced by HMW. It is defined analogously to common knowledge except that $E_{\mathcal{S}}\phi$ is replaced by $\boxdot E_{\mathcal{S}}\phi$.
        - HMW provides a characterization of optimality w.r.t. full-information exchange in terms of continual common knowledge.

Implementing P[1]

The continual common knowledge characterization of optimality w.r.t. full-information introduced by HMW cannot be implemented efficiently in general.

[6] Y. Moses and M. R. Tuttle. 1988. Programming simultaneous actions using common knowledge. *Algorithmica 3* (1988), 121–169. https://doi.org/10.1007/BF01762112.

Implementing $\mathbf{P}^1$

The continual common knowledge characterization of optimality w.r.t. full-information introduced by HMW cannot be implemented efficiently in general.

- But $\mathbf{P}^1$ uses only common knowledge, not continual common knowledge
- $\mathbf{P}^1$ does have a polynomial-time implementation P^{fip}, which uses the compact communication-graph representation of [Moses and Tuttle, 1988].[6]

[6] Y. Moses and M. R. Tuttle. 1988. Programming simultaneous actions using common knowledge. *Algorithmica 3* (1988), 121–169. https://doi.org/10.1007/BF01762112.

Implementing $\mathbf{P}^1$

The continual common knowledge characterization of optimality w.r.t. full-information introduced by HMW cannot be implemented efficiently in general.

- But $\mathbf{P}^1$ uses only common knowledge, not continual common knowledge
- $\mathbf{P}^1$ does have a polynomial-time implementation P^{fip}, which uses the compact communication-graph representation of [Moses and Tuttle, 1988].[6]

This shows that there exists polynomial-time protocols for EBA that are optimal (in general).

[6] Y. Moses and M. R. Tuttle. 1988. Programming simultaneous actions using common knowledge. *Algorithmica 3* (1988), 121–169. https://doi.org/10.1007/BF01762112.

Limited-information vs. full-information exchange

Total number of bits communicated:

- $P^{\min}$: n^2
- P^{basic}: $O(n^2 t)$
- P^{fip}: $O(n^4 t^2)$

Limited-information vs. full-information exchange

Total number of bits communicated:

- $P^{\min}$: n^2
- P^{basic}: $O(n^2 t)$
- P^{fip}: $O(n^4 t^2)$

Decision times in a failure-free run:

- If $\exists 0$ in the run, then all agents decide by round 2 in $P^{\min}, P^{basic}, P^{fip}$.
- If $\neg\exists 0$ in the run, then all agents decide by:
  - round $t + 2$ in $P^{\min}$
  - round 2 in P^{basic}, P^{fip}

Given the tradeoffs, P^{basic} might be preferable to P^{fip}.

Further work

- Characterizing optimality with respect to an information-exchange $\mathcal{E}$.
- Finding other optimal protocols for EBA under omission failures.
- Investigating optimality w.r.t. limited-information exchange in other settings.