APPENDIX A
SPECIFICATION

A. Matching

We can match two offers if they satisfy the following conditions: Job offer limit variables must be lower than the resource offer cap variables. Job offer max variables must be higher than the resource offer price variables. Additionally, there should be a mediator with the same architecture of JO which is trusted by both JC and RP.

$$RO.\textit{instruction\_capacity} \geq JO.\textit{instructions\_limit} \tag{6}$$
$$RO.\textit{ram\_capacity} \geq JO.\textit{ram\_limit} \tag{7}$$
$$RO.\textit{local\_storage\_capacity} \geq JO.\textit{local\_storage\_limit} \tag{8}$$
$$RO.\textit{bandwidth\_capacity} \geq JO.\textit{bandwidth\_limit} \tag{9}$$
$$RO.\textit{instruction\_price} \leq JO.\textit{instruction\_max\_price} \tag{10}$$
$$RO.\textit{bandwidth\_price} \leq JO.\textit{max\_bandwidth\_price} \tag{11}$$
$$JO.\textit{architecture} = RP.\textit{architecture} \tag{12}$$
$$JO.\textit{directory} \in RP.\textit{trusted\_directories} \tag{13}$$

$$\begin{aligned} \exists M: \ & M \in (JC.\textit{trusted\_mediators} \cap RP.\textit{trusted\_mediators}) \\ & \wedge M.\textit{architecture} = RP.\textit{architecture} \qquad (14) \\ & \wedge JO.\textit{directory} \in M.\textit{trusted\_directories} \qquad (15) \end{aligned}$$

$$\textit{current\_time} + RP.\textit{time\_per\_instruction} \cdot JO.\textit{instruction\_limit} \leq JO.\textit{completion\_deadline} \tag{16}$$

B. Payment

When the JC or the RP wants to post an offer for a job or resource, it will pay a deposit value to prevent it from cheating on the platform. This deposit value is a function of the posted offer which is more than the price of the job plus mediation. As the price of job or mediation is unclear at the time of posting the offer, a static $penalty\_rate \gg 1$ is applied as security deposit.

After the job is finished and both the JC and RP agree on the outcome, JC will receive the deposit minus the cost of the job and the RP will receive the deposited value plus the cost of the job.

In case of disagreement, the match will go for mediation. The mediators have a fixed price for their resources. After the submission of their verdict, the deposit of the party who is at fault will be forfeited in favor of the winner and winner will receive both the deposits minus the cost of mediation.

$$JO.\textit{deposit} = (\textit{instruction\_limit} \cdot \textit{instruction\_max\_price} + \textit{bandwidth\_limit} \cdot \textit{bandwidth\_max\_price}) \cdot (\theta + n) + \pi_a \tag{17}$$

$$RP.\textit{deposit} = (\textit{instruction\_cap} \cdot \textit{instruction\_price} + \textit{bandwidth\_cap} \cdot \textit{bandwidth\_price}) \cdot (\theta + n) + \pi_a \tag{18}$$

$$\begin{aligned} \pi_c = & \textit{result.instruction}_c\textit{ount} \cdot \textit{resource\_offer.instruction\_price} + \\ & \textit{result.bandwidth\_usage} \cdot \textit{resource\_offer.bandwidth\_price} \end{aligned} \tag{19}$$

APPENDIX B
DATA STRUCTURES

The platform is a composition of a smart contract, resource providers, job creators, Mediators, and Directories.

```
Platform {
   mediators: Mediator[],
   resource_providers: ResourceProvider[],
   job_creators: JobCreator[],

   resource_offers: ResourceOffer[],
```

```
    job_offers: JobOffer[],
    matches: Match[],
    results: JobResults[],
    penaltyRate : uint
}
```

All information is stored within well-known storage services. The platform itself is agnostic to a particular service. However, it is assumed that all storage providers, which we call Directories, support SFTP/SCP Gateways for uploading and downloading documents. To use SFTP, the client must authenticate itself and then upload or download files using fully qualified name. A file once uploaded can be shared with other users if their user-id is known. The platform can support multiple architectures. However, initially it will only have amd64 and armv7.

```
enum Architecture {
    amd64,
    armv7
}
```

A. Entities

```
JobCreator {
    trusted_mediators: Mediator[]
}

ResourceProvider {
    trusted_mediators: Mediator[],
    trusted_directories: address[],
    arch: Architecture,
    time_per_instruction: uint
}
```

`trusted_directories` lists Directory entities which the Resource Provider can use for both download and upload (the latter typically requires having some account on the Directory). `time_per_instruction` is an estimate of how much time the Resource Provider needs on average to execute an instruction, which is used when matching to check if the Resource Provider could finish the job before the deadline.

1) Mediators: Mediators are used for mediation. The mediator structure looks very much like the resource provider, except that it includes additional costs for mediation.

```
Mediator {
    supported_arch: Architecture,

    instruction_price: uint,
    bandwidth_price: uint,
    docker_bandwidth_price: uint
}
```

B. Offers

```
JobOffer {

        jobHash: hash,

        job_creator: JobCreator,

        URI: string,
        arch: Architecture,

        cpu_limit: uint,
        ram_limit: uint,
        local_storage_limit: uint,
        bandwidth_limit: uint,
```

```
    instruction_max_price: uint,
    max_bandwidth_price: uint,

    completion_deadline: datetime

    deposited_value: uint,

}
```

URI gives the location of the job-specific files on the Directory. The limits specify how many instructions the Job Creator is willing to pay for (after executing this many, the Resource Provider can give up and still get paid), how much RAM, local storage, and bandwidth the Resource Provider may have to use (again, after reaching these limits, the Resource Provider may stop). instruction_max_price specifies the maximum price per instruction that the Job Creator accepts, max_bandwidth_price specifies the maximum price per downloaded / uploaded byte (for the job execution layer, not for the base layer) that the Job Creator accepts. max_docker_bandwidth_price specifies the maximum total price that the Job Creator is willing to pay to the Resource Provider for downloading images that the Resource Provider does not have. time_to_completion is the deadline of RP for submitting the solution. If it happens that the RP has missed the deadline, the JC can call a timeout function which will punish the RP by taking all of it's deposit and giving it to the JC. Before an offer is matched, the participant can cancel the offer.

```
ResourceOffer {

    cpu_price: uint,
    cpu_cap: uint,

    memory_cap: uint,
    local_storage_cap: uint,

    bandwidth_cap: uint,
    bandwidth_price: uint,

    deposit_value: uint
}
```

Prices are per instruction or per byte. Capacities specify what resources the Resource Provider has, and they are used as constraints for matching. All of the participants can deposit as much as they want as long as it is more than the required JO/RP.deposit. deposit_value is the variable that holds the amount of deposited value for each offer.

C. Results

```
JobResult {
    match: Match,
    status: ResultStatus,
    URI: string,
    instruction_count: uint,
    bandwidth_usage: uint,
    timestamp: datetime
}
```

URI gives the location of the result on the Directory. instruction_count is the number of instruction that were executed by the ResourceProvider. bandwidth_usage is the number of bytes downloaded / uploaded by the Resource Provider for the job (not counting the download of Docker layers). The JC has a specific deadline for responding to a result. Whether to approve or decline it. If the deadline is missed the RP can accept the result instead of the JC.

```
MediatorResult {
    status: ResultStatus,
    uri: string,
```

```
    matchId: uint,

    hash: uint,

    instructionCount: uint,
    bandwidthUsage: uint,

    verdict: Verdict,
    faultyParty: Party
}
```

The platform on itself cannot determine which party is cheating merely based on the `hash` of the Mediator's Results. For example, the smart contract can not check whether the job was correctly posted to the directory by the Job Creator or the result was correctly posted to directory by the Resource Provider. Therefore, it is the responsibility of the Mediator to decide who should be punished in the ecosystem. `verdict` is the reason for deciding on the cheating party, and `faultyParty` is the cheating party.

```
Match {
    resource_offer: ResourceOffer,
    job_offer: JobOffer,
    mediator: Mediator
}

enum ResultStatus {
    Completed,
    Declined,
    JobDescriptionError,
    JobNotFound,
    MemoryExceeded,
    StorageExceeded,
    InstructionsExceeded,
    BandwidthExceeded,
    ExceptionOccured,
    DirectoryUnavailable
}
```

`Completed` means that the Resource Provider finished the job successfully and posted the results on the Directory. `JobDescriptionError` means that there is an error in the job description. `MemoryExceeded`, `InstructionsExceeded`, `BandwidthExceeded` mean that the job exceeded the limits specified in the JobOffer. `ExceptionOccured` means that an exception was encountered while executing the job, while `DirectoryUnavailable` means that the Resource Provider is unable to post results because the Directory is unavailable.

```
enum Verdict {
    ResultNotFound,
    TooMuchCost,
    WrongResults,
    CorrectResults,
    InvalidResultStatus
}
```

This is the reason that the Mediator chose to punish an actor. `ResultNotOnDirectory` means that the Resource Provider did not put the results in the directory. `TooMuchCost` means that the Resource Provider charged too much for the job. `WrongResults` means that the Resource Provider provided wrong results. `CorrectResults` means that the Resource Provider completed the job correctly and the JobCreator should be punished by sending the result for mediation. `InvalidResultStatus` means that the Resource Provider's mentioned ResultStatus is wrong. For example, it said that the job description was invalid and returned `JobDescriptionError`, but the description was correct.

```
enum Party {
    ResourceProvider,
```

```
        JobCreator
    }
```

These are the trustless parties in the ecosystem. Mediators should specify who cheated in a job and should be punished by specifying one of these parties.

D. Smart Contract Functions

TABLE V
FUNCTIONS PROVIDED BY THE SMART CONTRACT AND THEIR CORRESPONDING EVENTS, AS WELL AS THE AGENTS WHO CALL THE FUNCTIONS AND THE AGENTS THAT SUBSCRIBE TO THOSE EVENTS.

Function	Event	Caller	Subscriber
`setPenaltyRate`	`PenaltyRateSet`	Deployment	NA
`setReactionDeadline`	`ReactionDeadlineSet`	Deployment	NA
`registerMediator`	`MediatorRegistered`	M	M,S,JC,RP
`mediatorAddTrustedDirectory`	`MediatorAddedTrustedDirectory`	M	M,S
`mediatorAddSupportedFirstLayer`	`MediatorAddedSupportedFirstLayer`	M	M,S
`registerResourceProvider`	`ResourceProviderRegistered`	RP	RP,S
`resourceProviderAddTrustedMediator`	`ResourceProviderAddedTrustedMediator`	RP	RP,S
`resourceProviderAddTrustedDirectory`	`ResourceProviderAddedTrustedDirectory`	RP	RP,S
`resourceProviderAddSupportedFirstLayer`	`ResourceProviderAddedSupportedFirstLayer`	RP	RP,S
`registerJobCreator`	`JobCreatorRegistered`	JC	JC,S
`jobCreatorAddTrustedMediator`	`JobCreatorAddedTrustedMediator`	JC	JC,S
`postResOffer`	`ResourceOfferPosted`	RP	RP,S
`postJobOffer`	`JobOfferPosted`	JC	JC,S
`cancelJobOffer`	`JobOfferCanceled`	JC	JC,S
`cancelResOffer`	`ResourceOfferCanceled`	RP	RP,S
`postMatch`	`Matched`	S	S,RP
`postResult`	`ResultPosted`	RP	RP,JC
`rejectResult`	`ResultReaction,JobAssignedForMediation`	JC	JC,M,RP
`acceptResult`	`ResultReaction,MatchClosed`	JC	JC,RP
`postMediationResult`	`MediationResultPosted,MatchClosed`	M	M,JC,RP
`timeout`	`MatchClosed`	JC	JC,RP

APPENDIX C
PROOFS: EQUILIBRIUM ANALYSIS

The RP chooses between executing a job and attempting to deceive the JC. The JC chooses between verifying the result and accepting it without verification. Below we examine the payoffs for each combination to determine which conditions will cause each agent to always choose one action (the pure strategy) or to mix randomly between them. The utilities for the RP and JC can be found in Tables II and III respectively. We first evaluate the pure strategies for each agent, then evaluate their mixed strategies. We then assess how the platform parameters can affect the strategies of the agents. Finally we suggest parameter settings for the platform.

1) Resource Provider: The payoffs for the RP can be seen in Table II. We evaluate whether the RP prefers *execute* or *deceive* if it knows what the JC will choose (*verify* or *pass*) by first simplifying the payoff matrix. We do this by removing terms that are common to both options, leaving only the terms that distinguish them; the result is in Table VI.

According to Table VI if the JC chooses to pass, the RP will always choose to deceive since $c_e > c_d$. If the JC chooses to verify, the RP's optimal choice depends on the specific parameters. If $p_a^{n+1}\pi(\theta+n+1) > c_e - c_d$ then the RP will choose to execute, otherwise it will choose to deceive. Noting from our system constraints in Table I that $n > 0$ we know that $2\pi_c < \pi_c(\theta+n+1)$ and that $c_e < \pi_c$. The RP will choose to execute when $U_{DV}^{RP} < U_{EV}^{RP}$. Thus we find that the RP will execute when:

$$\left\{p_a^{n+1}\pi_c(\theta+n+1) > p_a^{n+1}2\pi_c\right\} > \left\{\pi_c > c_e > c_e - c_d\right\}$$
$$\text{which simplifies to} \quad p_a^{n+1}2\pi_c > \pi_c \text{ then to } p_a^{n+1} > \frac{1}{2}$$

TABLE VI
SIMPLIFIED RP PAYOFFS TO ASSESS DOMINANT STRATEGY WITH $\pi_d = \pi_c$. DOMINANT STRATEGIES ARE STARRED (*) FOR WHEN $p_a^{n+1} > \frac{1}{2}$.

	verify	pass
execute	$p_a^{n+1}\pi(\theta+n+1) - c_e^*$	$-c_e$
deceive	$-c_d$	$-c_d^*$

TABLE VII
SIMPLIFIED JC PAYOFFS TO ASSESS DOMINANT STRATEGY WITH $\pi_d = \pi_c + g_m$

	verify		pass
execute	$-\pi_c\,(1-p_a)\,(1-p_a^n)\,(n+\theta+1)$ $(1-p_a)\,(-g_m+2\pi_c)$	+	c_v
deceive	$2\pi_c - g_m - \pi_c\,(1-p_a^n)\,(n+\theta+1)$		c_v

This means that as long as $p_a^{n+1} > \frac{1}{2}$ and the JC is verifying the RP will choose to execute. We show that this bound can be set in Section C-A1.

2) Job Creator: The payoffs for the JC can be seen in Table III. We evaluate whether the JC chooses to verify or pass for each RP choice to execute or deceive.

If the RP chooses to execute, the JC will choose to verify if:

$$2\pi_c(1-p_a) - \Big(g_m + \pi_c(\theta+n+1)\Big)(1-p_a-p_a^n+p_a^{n+1}) > c_v \tag{20}$$

If the RP chooses to deceive, the JC will choose to verify if:

$$2\pi_c - \Big(g_m + \pi_c(\theta+n+1)\Big)(1-p_a^n) > c_v \tag{21}$$

In both instances, the JC will not verify if c_v is too large. If $c_v > 2\pi_c$ the JC will never verify. This is of particular interest for honest JC's which have $p_a = 1$ and $c_v > 0$. In this case: if the RP chooses to execute (see Eq. (20)) the JC will never verify, and if the RP chooses to pass the JC will verify unless $c_v > 2\pi_c$.

3) Pure Strategy Evaluation: Again referencing Tables II and III, the combination of choices {execute,pass} is not valid as a pure strategy since the the RP will always choose to deceive. If $p_a^{n+1} > \frac{1}{2}$ then {deceive,verify} is also invalid because RP will choose to execute. We show in Section C-A1 that the platform can set its parameters to enforce a lower bound on the JC's optimal value for p_a. If Eq. (20) is true then {execute,verify} is a valid pure strategy. {deceive, pass} is a pure strategy if Eq. (21) is false. However if Eq. (21) is true and Eq. (20) is false then there is no pure strategy equilibrium.

A. Mixed Strategy Nash Equilibrium

The JC will only verify all jobs if the cost of verification c_v is sufficiently low (0). Otherwise it may get better utility by adopting a mixed strategy, choosing to verify with probability p_v and choosing to pass with probability $1-p_v$. However if the JC passes, the RP will prefer to deceive (assuming that $p_a^{n+1} > \frac{1}{2}$) and so will also choose to adopt a mixed strategy where it executes with probability p_e and deceives with probability $1-p_e$.

$$\begin{aligned} p_e \cdot U_{EV}^{JC} + (1-p_e)\cdot U_{DV}^{JC} &= p_e \cdot U_{EP}^{JC} + (1-p_e)\cdot U_{DP}^{JC} \\ \text{Solve for } p_e; \quad p_e &= \frac{2\pi_c - c_v - g_m - \pi_c\,(1-p_a^n)\,(n+\theta+1)}{p_a\,(2\pi_c - g_m - \pi_c\,(1-p_a^n)\,(n+\theta+1))} \end{aligned} \tag{22}$$

To simplify our discussion we will replace the term $(1-p_a^n)\,(g_m + \pi(\theta+n+1))$ with A. In the denominator in Eq. (22), $A \neq 2\pi_c$, otherwise the denominator is 0 and the probability is invalid. While $2\pi_c - c_v < A < 2\pi_c$ the numerator is negative, but the denominator is still positive, resulting in a negative probability which is also invalid.

$$\begin{aligned} p_v \cdot U_{EV}^{RP} + (1-p_v)\cdot U_{EP}^{RP} &= p_v \cdot U_{DV}^{RP} + (1-p_v)\cdot U_{DP}^{RP} \\ \text{Solve for } p_v; \quad p_v &= \frac{c_e - c_d}{p_a^{n+1}\pi_c\,(\theta+n+1)} \end{aligned} \tag{23}$$

This value for p_v will be valid as long as $p_a \neq 0$ and the numerator is less than the denominator, since $0 \leq p_v \leq 1$. The numerator is always positive by our assumption that $c_e > c_d$, and the denominator is also always positive.

The JC will choose a mixed strategy if mixing results in a better utility than a pure strategy. Eq. (26) shows the total utility of the JC. If $U^{JC} > \max\left(U_{EV}^{JC}, U_{DV}^{JC}, U_{EP}^{JC}, U_{DP}^{JC}\right)$ then the JC will mix.

$$U_V^{JC} = p_e \cdot U_{EV}^{JC} + (1-p_e)\cdot U_{DV}^{JC} \tag{24}$$

$$U_P^{JC} = p_e \cdot U_{EP}^{JC} + (1-p_e)\cdot U_{DP}^{JC} \tag{25}$$

$$U^{JC} = p_v \cdot U_V^{JC} + (1-p_v)\cdot U_P^{JC} \tag{26}$$

TABLE VIII
HOW THE OPTIMAL VALUE FOR p_a VARIES WITH AN INCREASE IN EACH OF THE PARAMETERS.

parameter action	p_a
n increase	increase
θ increase	increase
g_m increase	increase
π_c increase	decrease up to limit
p_e increase	decrease

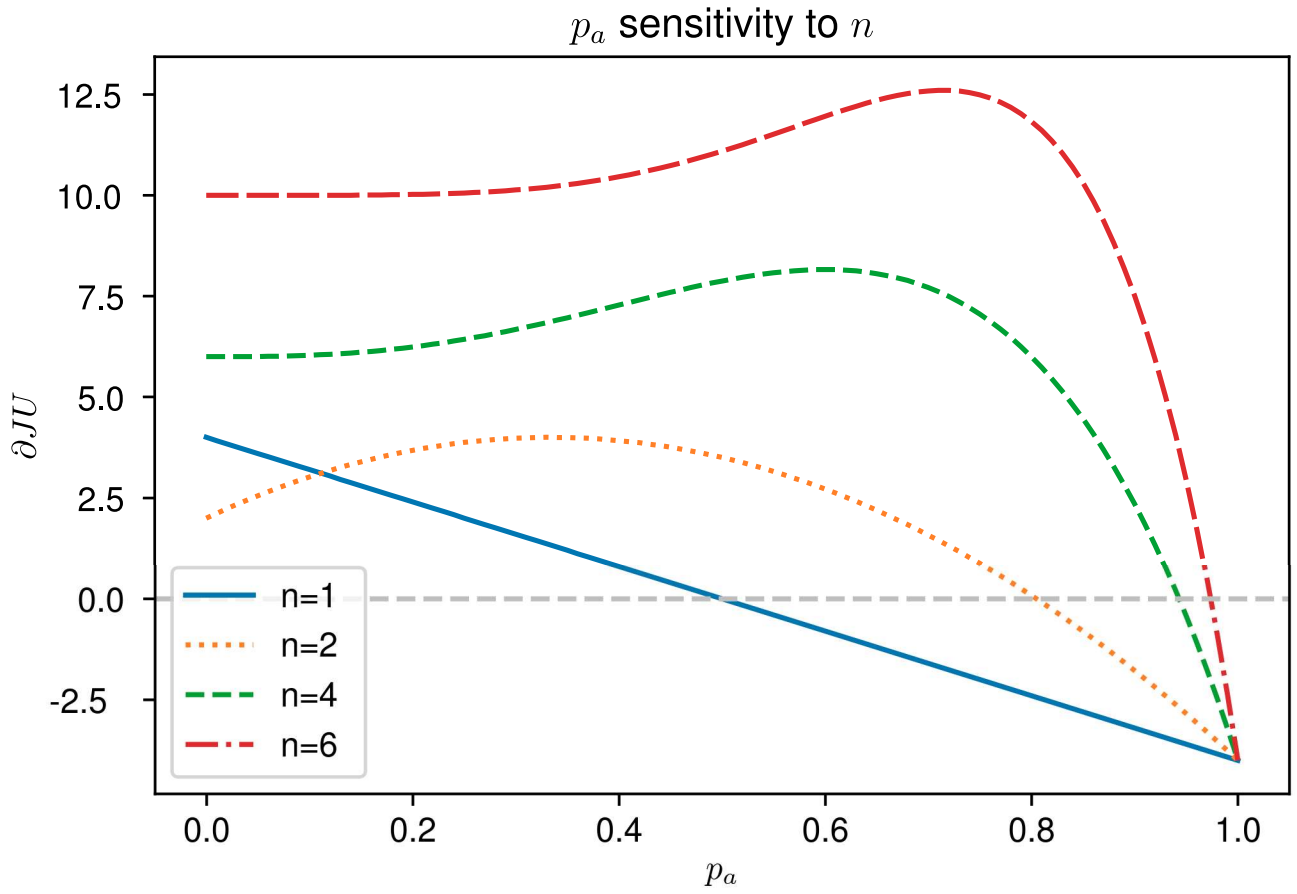

Fig. 9. In this plot we vary the value of n and plot p_a against $\frac{\partial U^{JC}}{p_a}$. This shows that as n increases so does the optimal value of p_a. For this plot $\pi_c = 2$, $g_m = 0$, $\theta = 0$, $c_v = 1$, $b = 4$, $p_e = 1$

1) Bounding p_a: We assume that the JC is a rational actor, and will choose p_a to optimize its utility. In this section we show that the platform can enforce a lower bound on the optimal value for p_a by setting n and θ. We do this by taking the derivative of the utility function with respect to p_a and evaluate how variation in each of the parameters impacts the optimal value for p_a. The derivative of the JC utility is Eq. (27) which we set to 0 and rearrange terms to obtain Eq. (28) facilitating the parameter exploration. Table VIII summarizes how p_a varies with an increase in each of the parameters.

$$\frac{\partial}{p_a} U_{EV}^{JC} = \Big(g_m + \pi_c(\theta + n + 1)\Big)\left(p_a^n(-np_e + \frac{n}{p_a} - p_e) + p_e\right) - 2\pi_c p_e \tag{27}$$

$$\frac{2\pi_c}{g_m + \pi_c(\theta + n + 1)} = 1 - p_a^n + \frac{np_a^{n-1}}{p_e} - np_a^n \tag{28}$$

If π_c increases, the left-hand side (lhs) increases and approaches $\frac{2}{\theta+n+1}$, then the right-hand side (rhs) must also incrase meaning that p_a must decrease.

If $g_m = 0$ π_c has no impact on p_a and will be the smallest possible p_a achievable by modulating π_c

If g_m increases lhs decreases meaning p_a will increase.

If θ increases lhs decreases meaning p_a will increase.

If n increases p_a increases, see Fig. 9.

Now that we know how each parameter impacts the optimal value for p_a we can reason about how the platform parameters can be used to set its lower bound. We choose the worst case values for each of the parameters, $g_m = 0$[3], $p_e = 1$, $n = 1$, $\theta = 0$. The parameters b and c_v have no impact on the optimal value for p_a. Fig. 9 uses these parameters and we see that when $n = 1$ the optimal $p_a = 0.5$ and when $n = 4$ the optimal $p_a = .943$. Thus we see that setting n can set a lower bound on the optimal p_a. θ also can adjust p_a. To minimize the cost of replication we choose $p_a = 2$ and set $\theta = 50$ which yields a minimum $p_a = .99$. The RP can use this bound to compute a pessimistic p_e with Eq. (22), setting $c_v = 0$. Similarly the JC can set a pessimistic p_v with equation Eq. (23), setting $c_e = \pi_c$ and $c_d = 0$. It is the responsibility of the JC and RP to construct offers such that if they follow the dominant strategy their utility is positive.

APPENDIX D
FUNCTION GAS COSTS

$$\begin{aligned} \text{nominal cost} &= \texttt{postjobOfferPartOne} + \texttt{postjobOfferPartTwo} + \texttt{acceptResult} \\ \text{nominal cost} &= 2.36E5 + 2.16E5 + 1.40E5 \\ \text{nominal cost} &= 5.92E5 \end{aligned} \tag{29}$$

TABLE IX
GAS COST FOR EACH PLATFORM FUNCTION CALL, MEASURED UPON RECEIPT OF EVENT EMITTED BY THAT FUNCTION. GASESTIMATE IS THE VALUE RETURNED WHEN CALLING THE ETH_ESTIMATEGAS METHOD

Function	mean	std.dev.	gasEstimate
setPenaltyRate	4.34E+04	0.00E+00	2.21E+04
setReactionDeadline	4.31E+04	0.00E+00	2.16E+04
acceptResult	1.40E+05	6.47E+04	6.29E+04
postResult	1.91E+05	1.60E+04	2.25E+05
postMatch	1.35E+05	1.70E+03	1.27E+05
postJobOfferPartOne	2.36E+05	4.59E+03	2.28E+05
postJobOfferPartTwo	2.16E+05	0.00E+00	2.27E+05
postResOffer	2.42E+05	2.16E+03	2.32E+05
registerMediator	1.37E+05	0.00E+00	8.51E+04
registerResourceProvider	8.12E+04	0.00E+00	2.54E+04
resourceProviderAddTM	6.50E+04	0.00E+00	4.23E+04
registerJobCreator	3.40E+04	0.00E+00	3.20E+03
jobCreatorAddTM	6.53E+04	0.00E+00	4.26E+04
postMediationResult	1.87E+05	1.56E+04	2.40E+05
rejectResult	5.39E+04	9.90E+00	4.71E+04
nominal cost	5.92E+05		

[3] p_c cancels in this case