Pricing Non-fungible Resources

Toward Multi-dimensional Fee Markets

Theo Diamandis
Bain Capital Crypto & MIT

Joint work with Alex Evans, Tarun Chitra, Guillermo Angeris
Fee markets with a joint unit of account are inefficient.

Our work: framework to optimally set multi-dimensional fees.
Why are transactions so expensive?
Fixed unit of account leads to DoS attacks

- All opcodes have fixed prices relative to each other
- Potential mismatch between relative prices & resource usage leads to resource exhaustion attacks (DoS attacks)
  - EXTCODESIZE attack in 2016 exploited disk read mispricing
  - Opcode costs manually adjusted (EIP-150)
Fixed unit of account limits throughput

Mempool
util = 4
CPU
bandwidth

util = 2
CPU
bandwidth

1d market
($gas = 3)$

CPU
bandwidth

CPU
bandwidth

CPU
bandwidth

2d market
($CPU = 3, BW = 1)$

CPU
bandwidth

CPU
bandwidth

CPU
bandwidth

CPU
bandwidth
Orthogonal resources should be priced separately.

We need a mechanism for resource price discovery!
But what is a resource?

- Anything that can be **metered**
  - Blobs (EIP-2242 & EIP-4844)
  - Compute, memory, storage
  - Opcodes
  - Sequences of opcodes
  - Compute on a specific core
  - ...
Let’s formalize this

- A transaction \( j \) consumes a vector of resources \( a_j \in \mathbb{R}^m \)
  - \( (a_j)_i = \text{amt of resource } i \text{ consumed by tx } j \)
Let’s formalize this

- A transaction $j$ consumes a vector of resources $a_j \in \mathbb{R}^m$
  - $(a_j)_i =$ amt of resource $i$ consumed by tx $j$
- $x \in \{0, 1\}^n$ records which of $n$ possible tx’s included in block
  - $x_j = 1$ if tx $j$ included, 0 otherwise
Let’s formalize this

- A transaction $j$ consumes a vector of resources $a_j \in \mathbb{R}^m$
  - $(a_j)_i =$ amt of resource $i$ consumed by tx $j$
- $x \in \{0, 1\}^n$ records which of $n$ possible tx’s included in block
  - $x_j = 1$ if tx $j$ included, 0 otherwise
- The quantity of resources consumed by this block is then

$$y = \sum_{j=1}^{n} x_j a_j = Ax$$
We constrain & charge for each resource

- Define a resource target $b^★$
  - Deviation from the target is $Ax - b^★$
  - In Ethereum: $b^★ = 15\text{M gas}$
We constrain & charge for each resource

- Define a resource target $b^*$
  - Deviation from the target is $Ax - b^*$
  - In Ethereum: $b^* = 15M \text{ gas}$

- Define a resource limit $b$
  - Tx included must satisfy $Ax \leq b$
We constrain & charge for each resource

- Define a resource target $\mathbf{b}^\star$
  - Deviation from the target is $\mathbf{A}\mathbf{x} - \mathbf{b}^\star$
  - In Ethereum: $\mathbf{b}^\star = 15\text{M gas}$

- Define a resource limit $\mathbf{b}$
  - Tx included must satisfy $\mathbf{A}\mathbf{x} \leq \mathbf{b}$

- Charge for each resource usage (EIP-1559)
  - $\mathbf{p} \rightarrow$ transaction $\mathbf{j}$ costs $\mathbf{p}^T\mathbf{a}_j = \Sigma_i p_i(a_{j,i})$
But how do we determine prices?

- We want a few properties:
  - $(Ax)_i = b^*_i \rightarrow$ no update
  - $(Ax)_i > b^*_i \rightarrow p_i$ increases
  - $(Ax)_i < b^*_i \rightarrow p_i$ decreases
But how do we determine prices?

- We want a few properties:
  - \((Ax)_i = b^*_i\) → no update
  - \((Ax)_i > b^*_i\) → \(p_i\) increases
  - \((Ax)_i < b^*_i\) → \(p_i\) decreases

- Proposal

\[
p_{i}^{k+1} = p_{i}^{k} \cdot \exp (\eta (Ax - b^*)_i)
\]

Is this a good update rule?
Update rules are implicitly solving an optimization problem.

Specific choice of objective by network designer → rule.
The resource allocation problem
Setting (for now):

Network designer is omniscient & determines tx in each block.
Loss function is network designer’s ‘unhappiness’ w. resource utilization

\[ \ell(y) = \begin{cases} 
0 & y = b^* \\
\infty & \text{otherwise.} 
\end{cases} \]

\[ \ell(y) = \begin{cases} 
0 & y \leq b^* \\
\infty & \text{otherwise.} 
\end{cases} \]

Network designer determines loss function to define resource allocation problem.
We encode all tx constraints in set $S$

- $S \subseteq \{0, 1\}^n$ is the set of allowable transactions
  - Network constraints, e.g. $Ax \leq b$
  - Interactions among tx’s, e.g., bidders for MEV
We encode all tx constraints in set $S$

- $S \subseteq \{0, 1\}^n$ is the set of allowable transactions
  - Network constraints, e.g. $Ax \leq b$
  - Interactions among tx’s, e.g., bidders for MEV
- Consider the convex hull of $S$: $\text{conv}(S)$
  - $x_j \in (0, 1) \Rightarrow$ tx $j$ included after roughly $1/x_j$ blocks
Transaction $j$ included → user’s & validators’ joint utility is $q_j$

- Tx producers = users + validators
- We almost never know $q$ in practice
- However, the network does not need to know $q$!
The resource allocation problem:

\[
\begin{align*}
\text{maximize} & \quad q^T x - \ell(y) \\
\text{subject to} & \quad y = Ax \\
& \quad x \in \text{conv}(S).
\end{align*}
\]

- This is ‘best case’ scenario: tx’s included to maximize utility, BUT
  - Cannot be implemented–designer does not build blocks!
  - \( q \) is unknowable
  - Cannot partially include tx’s!
Setting prices via duality
Duality theory: relaxing constraints to penalties

\[
\begin{align*}
\text{maximize} & \quad q^T x - \ell(y) \\
\text{subject to} & \quad y = Ax
\end{align*}
\]

\[x \in \text{conv}(S).\]

- Network designer cares about throughput \( y \), based on inc. tx’s \( x \)
- We `decouple` utilization of network and that of tx producers
- Correctly set penalty \( \rightarrow \) the dual problem is equivalent to the original problem & these utilizations are equal
Dual decouples tx producers & network

\[
g(p) = \sup_y \left( p^T y - \ell(y) \right) + \sup_{x \in \text{conv}(S)} \left( (q - A^T p)^T x \right).
\]

- \( p \) is dual variable for constraint \( y = Ax \)
  - Relaxing constraint to penalty → pay per unit violation
- First term is easy to evaluate: can be done on chain!
- Let’s look at the second term…

Network Problem

Block Building Problem
Second term: block building problem

\[
\begin{align*}
\text{maximize} & \quad (q - A^T p)^T x \\
\text{subject to} & \quad x \in \text{conv}(S),
\end{align*}
\]

- Maximizing net tx utility subject to tx constraints
- Same optimal value as replacing \( \text{conv}(S) \) with \( S \)
- Solved by block producers! → Network can observe \( x \)
What do we get at optimality?

- Let $p^\star$ be a minimizer of $g(p)$ \([prices are set optimally]\)
- Assume the block building problem has optimal sol $x^\star$
- The optimality conditions are

$$\nabla g(p^\star) = y^\star - Ax^\star = 0$$

- Where $y^\star$ satisfies $\nabla y^\star = p^\star$
Prices that minimize $g$ charge the tx producers exactly the marginal costs faced by the network:

$$\nabla \ell(Ax^*) = p^*$$
And these prices incentivize tx producers to include tx’s that maximize welfare generated $q^T \cdot x$ minus the network loss $\mathcal{L}(y)$
Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

$$\nabla g(p) = y^*(p) - Ax^*(p)$$
Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

$$\nabla g(p) = y^*(p) - Ax^*(p)$$

- Network determines $y^*(p)$ (computationally easy)
Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

\[ \nabla g(p) = y^*(p) - Ax^*(p) \]

- Network determines $y^*(p)$ (computationally easy)

- $x^*(p)$ found by observing tx’s included in block by tx producers
Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

$$\nabla g(p) = y^*(p) - Ax^*(p)$$

- Network determines $y^*(p)$ (computationally easy)
- $x^*(p)$ found by observing tx’s included in block by tx producers
- Then apply any optimization method (e.g., gradient descent)

$$p^{k+1} = p^k - \eta \nabla g(p^k).$$
Some simple examples:

**Loss function**

\[
\ell(y) = \begin{cases} 
0 & y = b^* \\
\infty & \text{otherwise.}
\end{cases}
\]

**Update rule**

\[
p^{k+1} = p^k - \eta (b^* - Ax^*)
\]
Some simple examples:

\[ \ell(y) = \begin{cases} 
0 & y = b^* \\
\infty & \text{otherwise.} 
\end{cases} \]

\[ \ell(y) = \begin{cases} 
0 & y \leq b^* \\
\infty & \text{otherwise.} 
\end{cases} \]

**Loss function**

**Update rule**

\[ p^{k+1} = p^k - \eta (b^* - Ax^*) \]

\[ p^{k+1} = (p^k - \eta (b^* - Ax^*))_+ \]
1-dim prices hurt networks
Multidimensional fees increase throughput
Even when the tx distribution shifts
And resource utilization better tracks targets.
Future work & open questions

For researchers:
● What is the dynamical behavior? How do we make this strategy-proof?
  [Game-theoretic analysis of incentives]
● What update rules are most useful? [convergence behavior vs complexity]

For system designers:
● What should the resources be in a given system?
● How do you optimally trade-off between complexity & ease of use?
● How do you design a loss function for desired performance characteristics?
For more, check out our paper!

Thank you!

Theo Diamandis
Bain Capital Crypto & MIT

@theo_diamandis