Применение механизма аукционов и мультиагентных технологий для распределения ресурсов и задач в распределенной вычислительной системе

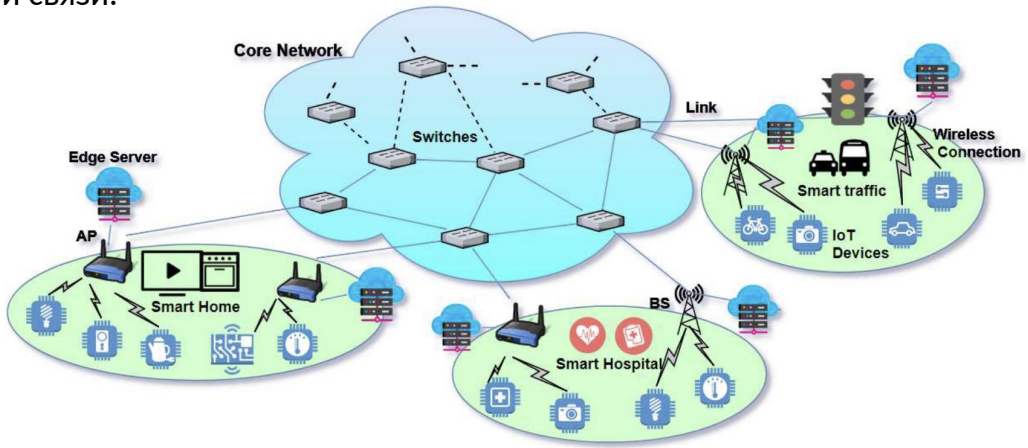
Терещенко Дмитрий Владиславович

инженер-исследователь кафедры компьютерного моделирования и многопроцессорных систем

Распределенные вычисления на IoT

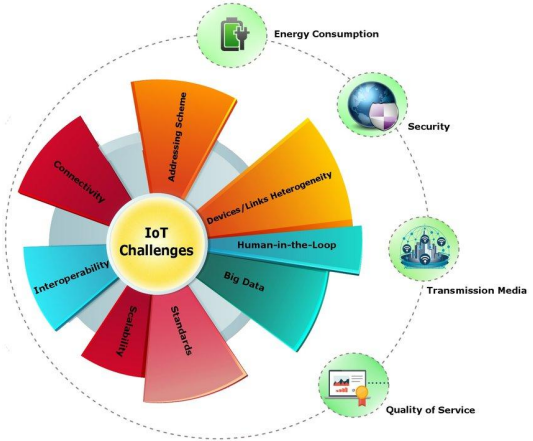
Интернет вещей (IoT) - это совокупность устройств и датчиков, которые собирают данные и обмениваются ими, при этом зачастую обладают ограниченными вычислительными ресурсами.

В основе распределенных вычислений на IoT лежит технология Edge вычислений, что позволяет устройствам обрабатывать данные локально. **Перенос вычислений на edge-устройства** (периферийные устройства) требуется для задач, где критически важны низкая задержка, локальная обработка данных, энергоэффективность или работа в условиях нестабильной связи.



Современные вызовы IoT

- **Автономность**. Способность принимать решения на месте (на edge-устройствах, без облачных вычислений).
- **Гетерогенность**. Учет неоднородности устройств (в частности, с точки зрения вычислительных возможностей).
- **Оптимизация QoS**. Например, сокращение времени на передачу данных в облако и обратно для оптимизации задержек.
- **Экономия трафика и сетевых ресурсов**. Сокращения сетевых коммуникаций между устройствами и облаком (например, отправка в облако только релевантной информации), что снижает нагрузку на сеть и стоимость передачи.
- **Устойчивость к сетевым сбоям**. Возможность функционирования даже при отсутствии связи с облаком (например, сохраняя результаты для последующей синхронизации).
- **Энергоэффективность**. Оптимальное использование энергии устройств на вычислениях и сетевой коммуникации.
- **Масштабируемость**. Возможность масштабировать систему без перегрузки центральных облачных серверов.



Цели

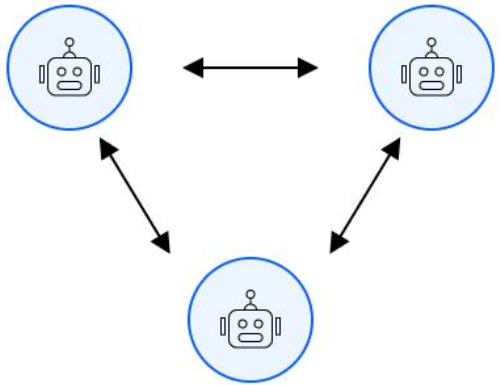
1. Разработать самоорганизующуюся вычислительную систему
2. Уйти от централизованного управления ресурсами
3. Достичь оптимальности, справедливости и высокой утилизации распределения ограниченных ресурсов
4. Учесть топологию сети устройств и задержки при коммуникациях

Мультиагентные системы

Мультиагентные системы (MAS) - системы, где знания, действия и контроль распределяются между агентами.

Агенты часто работают сообща и достигают своих общих и индивидуальных целей в условиях ограниченных ресурсов.

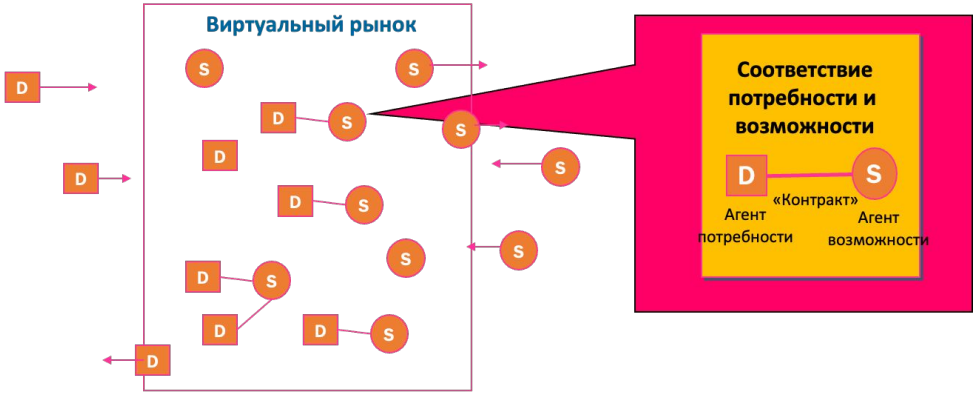
Ключевым механизмом являются **переговоры**, поскольку у агентов могут быть конфликтующие интересы или необходимость в сотрудничестве.



Виртуальный рынок

Мы рассматриваем проблему распределения ресурсов как **виртуальный рынок**, на котором агенты продавца (возможности) и покупателя (потребности) участвуют в аукционе. Сам аукцион проводит агент координатор.

Агенты обладают различными платежными возможностями, которые позволяют им действовать по-разному в зависимости от текущего контекста или ситуации на рынке.



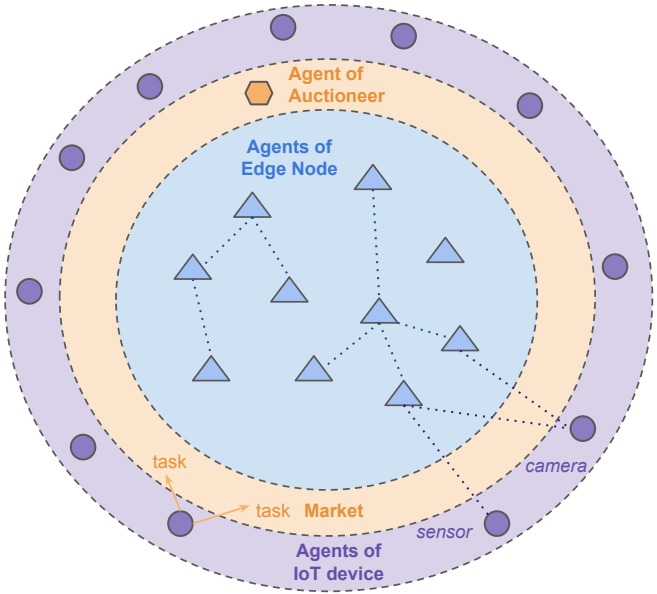
В.И. Городецкий, О.Н. Граничин, П.О. Скобелев. Децентрализация, самоорганизация и эмерджентный интеллект – цифровой взрыв умных технологий // Материалы общих заседаний 15-й Мультиконференции по проблемам управления, Санкт-Петербург, 04-06 октября 2022 г. – СПб.: АО «Концерн «ЦНИИ «Электроприбор», 2022. – С. 40-54.С.П.

Концептуальная архитектура системы

Многоагентная модель:

- **Агент Edge-узла** (поставщик ресурсов) / Агент возможности
- **Агент IoT-устройства** (потребитель, генерирует задачи) / Агент потребности
- **Агент Аукционера** (координирует распределение)

Граф сети: вершины — агенты, рёбра — каналы связи с задержками

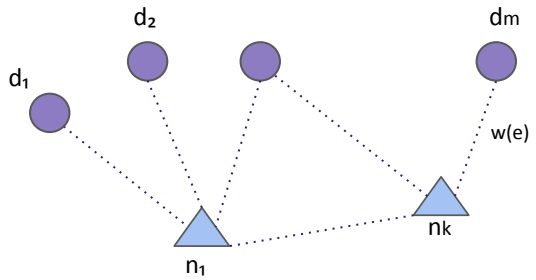


Модель системы

Узлы и топология сети

Пусть **G = (V, E)** - граф сети, где:

- **V = N ∪ D** - множество вершин, состоящее из:
  - **N = {n_1, n_2, ..., n_k}** - множество Edge-узлов (поставщики ресурсов)
  - **D = {d_1, d_2, ..., d_m}** - множество IoT-устройств (потребители ресурсов)
- **E** - множество рёбер с весами **w(e)** для каждого ребра **e ∈ E**, представляющими задержки передачи данных



Задачи

Каждое IoT-устройство **d_i ∈ D** генерирует задачи из множества **T_i = {t_{i1}, t_{i2}, ..., t_{il_i}}**. Каждая задача **t_{il_i} ∈ T_i** характеризуется:

- **cpu(t)** - требуемые вычислительные ресурсы
- **mem(t)** - требуемая память
- **data(t)** - объем данных задачи
- **deadline(t)** - крайний срок выполнения
- **priority(t)** - приоритет задачи

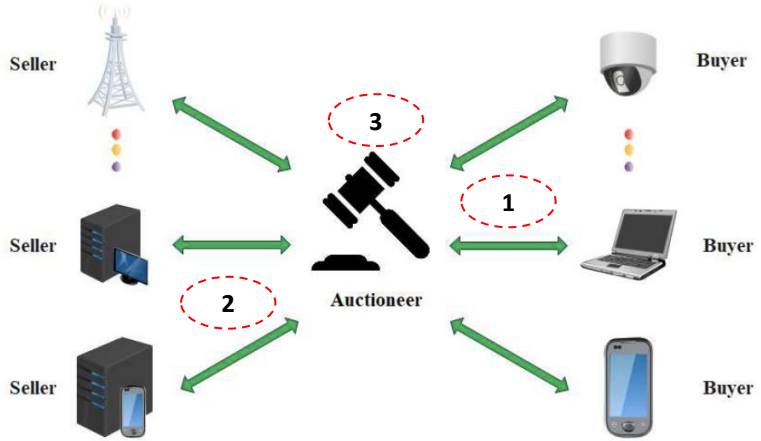
Ресурсы

Каждый Edge-узел **n_j ∈ N** обладает ресурсами:

- **C_j = {cpu_j, mem_j}** - доступные вычислительные ресурсы
- **A_j(t)** - доступные ресурсы в момент времени **t**

Протокол аукциона (переговоров)

1. **Фаза подачи заявок**: Агенты IoT-устройств формируют заявки на выполнение задач (task). В заявку входит необходимое кол-во ресурсов для задачи и объявленная ценность (полезность) от выполнения этой задачи
2. **Фаза оценки заявок**: Агенты Edge-узлов оценивают стоимость выполнения задач с учётом локальной загрузки и сетевой задержки
3. **Фаза определения победителя**: Агент Аукционера определяет победителя. Побеждает узел с минимальной стоимостью, выплата — механизм VCG платежей.



Любой агент может инициировать взаимодействие в соответствии с протоколом, взяв на себя соответствующую роль. В результате получаем назначение "один к одному" - одной задачи (или ресурса) одному подрядчику.

Функция полезности IoT-устройства

Основная формула*	$U_i(t, n_j, \tau) = V_i(t) \cdot \phi(\tau, deadline(t)) - E_i(t, n_j)$	Количественная оценка «выгоды» или «удовлетворенности» устройства от выполнения его задачи на edge-узле с учетом задержки и энергозатрат.
Базовая ценность задачи	$V_i(t) = priority(t) \cdot (cpu(t) + mem(t)) \cdot \beta_i$	Сколько «пользы» принесет устройству успешное выполнение задачи. Где • **βi** - коэффициент значимости для устройства **d_i**.
Функция штрафа за задержку	$\phi(\tau, deadline(t)) = \begin{cases} e^{-\alpha \cdot \frac{\tau}{deadline(t)}} & \text{если } \tau \le deadline(t) \\ 0 & \text{если } \tau > deadline(t) \end{cases}$	Насколько эта польза уменьшается из-за задержки. Где • **τ** - оценочное время выполнение задачи • **α > 0** - параметр чувствительности к задержке.
Функция энергозатрат	$E_i(t, n_j)$ = Энергозатраты на выполнение задачи (например, на передачу данных к edge-узлу и получение результата).	

* G. Li, J. Cai, X. Chen and Z. Su, "Nonlinear Online Incentive Mechanism Design in Edge Computing Systems With Energy Budget," in IEEE Transactions on Mobile Computing, vol. 22, no. 7, pp. 4086-4102, 1 July 2023, doi: 10.1109/TMC.2022.3148034.

Функция стоимости Edge-узла

Основная формула*	$C_j(t, d_i) = C_{comp}(t) + C_{comm}(t, d_i) + C_{load}(t)$	Оценка всех реальных затрат, которые несет edge-узел при выполнении задачи для IoT-устройства, выраженная в условных единицах.
Вычислительная стоимость	$C_{comp}(t) = p_{cpu} \cdot cpu(t) + p_{mem} \cdot mem(t)$	Отражает реальные издержки на вычисления. Где • p_{cpu} и p_{mem} - удельные стоимости CPU и памяти.
Коммуникационная стоимость	$C_{comm}(t, d_i) = \begin{cases} w(d_i, n_j) \cdot data(t) \cdot p_{net} & \text{если } (d_i, n_j) \in E \\ dist(d_i, n_j) \cdot data(t) \cdot p_{net} \cdot \gamma & \text{если } (d_i, n_j) \notin E \end{cases}$	Моделирует «цену» передачи данных. Где • $w(d_i, n_j)$ - прямой вес ребра между устройством и узлом • $dist(d_i, n_j)$ - кратчайшее расстояние в графе • p_{net} - стоимость единицы сетевого трафика • $\gamma > 1$ - штраф за непрямое соединение
Стоимость загрузки	$C_{load}(t) = C_{comp}(t) \cdot \left(1 + \left(\frac{Load_j}{Cap_j}\right)^2\right)$	Отражает влияние текущей загрузки. Где • **Load_j** и **Cap_j** - текущая загрузка и емкость узла **n□**.

* H. Qiu, K. Zhu, N. C. Luong, C. Yi, D. Niyato and D. I. Kim, "Applications of Auction and Mechanism Design in Edge Computing: A Survey," in IEEE Transactions on Cognitive Communications and Networking, vol. 8, no. 2, pp. 1034-1058, June 2022, doi: 10.1109/TCCN.2022.3147196.

Механизмы аукциона

Механизмы аукционов направлены на достижение баланса между максимизацией общего социального благополучия и удовлетворением интересов всех участников торгов.

Заявки	$b_i(t) = \max_{n_j \in N_{accessible}(d_i)} U_i(t, n_j, \tau_{i,j})$	Каждое IoT-устройство d_i для задачи **t** формирует заявку **bi(t)**. Где: • $N_{accessible(di)}$ - множество доступных edge-узлов для устройства d_i • **Ui** - выгода IoT-устройства d_i от назначения задачи t на edge-узел **nj**
Определение победителя	$n^* = \arg \min_{n_j \in N_{capable}(t)} C_j(t, d_i)$	Для каждой задачи **t** от устройства d_i аукционер определяет победителя. Где: • $N_{capable(t)}$ - множество узлов, способных выполнить задачу **t**. • **Cj** - стоимость edge-узла **nj** при выполнении задачи **t** от устройства **di**
Социальное благополучие*	$SW = \sum_{i=1}^{m} \sum_{t \in T_i} \sum_{j=1}^{k} x_{i,j}^t \cdot (b_i(t) - C_j(t, d_i))$	Целевая функция для максимизации суммы чистых выгод (выгода минус издержки) по всем участникам системы. Где: • Где $x_{ij}^t \in \{0,1\}$ - индикатор, если задача **t** от устройства d_i выполняется на узле **n□**.
Расчёт платежа (VCG)*	$p_i^{VCG}(t) = SW_{-i} - (SW - v_i(x_i^*))$	Сколько должен заплатить IoT-устройство, чтобы у агентов не было стимула лгать о своей полезности / стоимости. Где: • SW_{-i} - оптимальное социальное благополучие без участия устройства d_i • $v_i(x_i^*)$ - полученная ценность агентом d_i при оптимальном распределении

* Подходы для определения SW изучаются в Handbook of Social Choice and Welfare. Выбранный подход эквивалентен классическому cost-benefit analysis (CBA) [Matthew D. Adler].

* Механизм Викри-Кларка-Гроувса

Свойства решения

- **Оптимальность распределения:** Максимизация социального благополучия обеспечивает оптимальное распределение по Парето. Оптимально по Парето тогда и только тогда, когда невозможно (строго) улучшить индивидуальное благосостояние агента, не ухудшая положение других.
- **Устойчивость к стратегическому поведению**: Truth-revealing свойство VCG предотвращает манипуляции со стороны агентов.
- **Справедливость (Fairness) участия**: Участие в аукционе является выгодно всем агентам.
- **Эффективность в реальном времени**: Локальные аукционы обеспечивают быструю реакцию на изменения в системе.

Пример сценария: оптимизация трафика

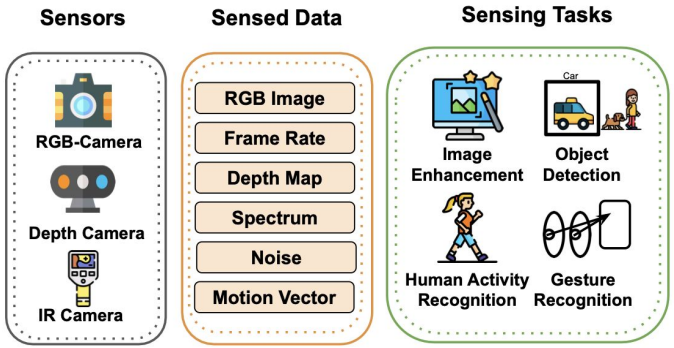
IoT устройства: Камеры и датчики фиксируют события на дороге.

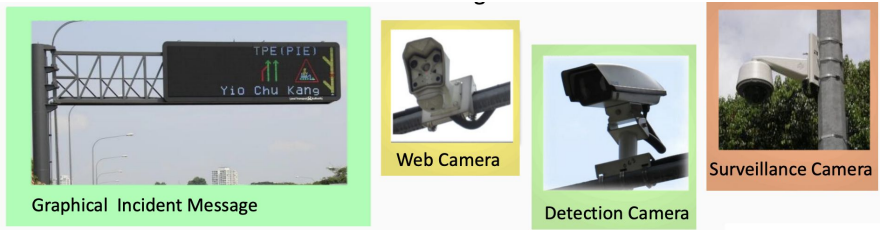
Задачи для Edge computing:

- Предобработка данных (сжатие видео, фильтрация шума).
- Распознавание ДТП по данным с видео.
- Прогнозирование заторов по данным с датчиков (например, уменьшение скорости потока движения).

Цель:

- Светофоры автоматически переключаются для перенаправления трафика.





Методология проведения экспериментов

Конфигурация системы:

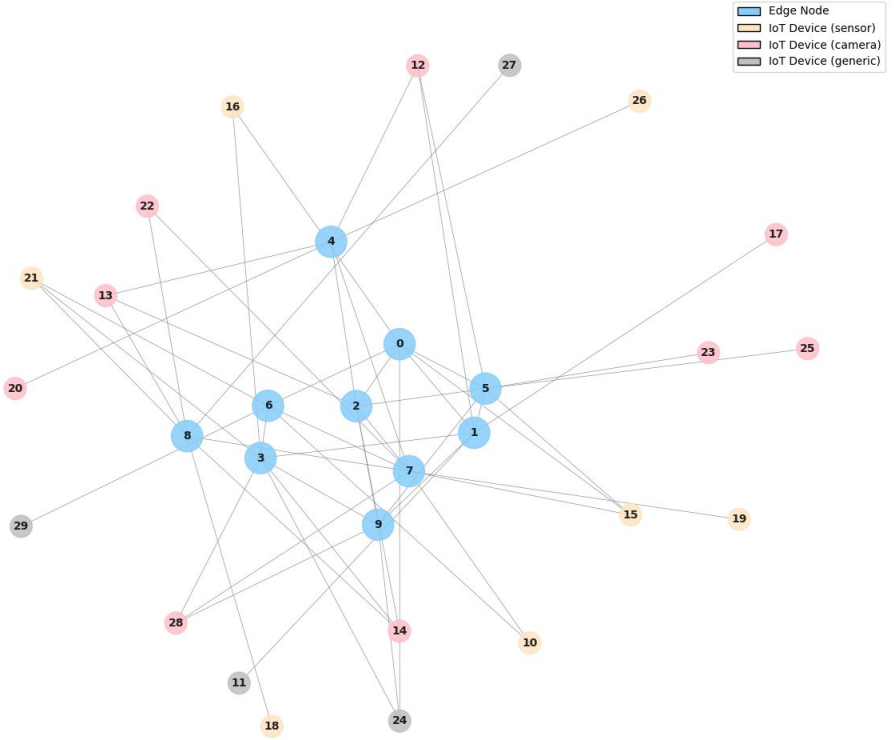
- Малый стенд: 10 edge-узлов, 20 IoT-устройств
- Средний стенд: 25 edge-узлов, 50 IoT-устройств
- Большой стенд: 50 edge-узлов, 100 IoT-устройств

Параметры узлов:

- CPU: 20-40 единиц вычислительных ресурсов
- Память: 32-64 ГБ
- Сетевые задержки: 0.5-5.0 мс между узлами
- Энергопотребление: 2.0 Дж на единицу CPU

Характеристики задач:

- Типы устройств: сенсоры (легкие задачи), камеры (тяжелые задачи), общие устройства
- Диапазон CPU: 0.1-8.0 единиц
- Диапазон памяти: 0.1-16.0 ГБ
- Сроки выполнения: 0.5-10.0 секунд



Метрики

1. **Оптимальность распределения (социальное благополучие)**: Фундаментальная метрика в теории механизмов, показывающая суммарную полезность всех участников системы.
2. **Эффективность утилизации ресурсов**: Отношение распределенных ресурсов к оставшимся. Позволяет выявить недоиспользование ресурсов.
3. **Справедливость распределения (индекс Джайна)**: Справедливость распределения ресурсов между агентами. Устраняет неравномерность и обеспечивает долгосрочную стабильность системы.
4. **Время конвергенции**: Время достижения равновесия в системе. Важно для real-time систем.

Почему индекс справедливости Джайна*?

$$J(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

Преимущества:

- Нормализован в диапазоне [1/n, 1]
- Чувствителен к неравенству в распределении
- Широко используется в сетевых системах

Альтернативы: Коэффициент Джини, энтропия Шеннона, но индекс Джайна проще в вычислении и интерпретации.

* Rezaeinia, N., Góez, J.C. & Guajardo, M. On efficiency and the Jain's fairness index in integer assignment problems. Comput Manag Sci 20, 42 (2023). https://doi.org/10.1007/s10287-023-00477-9

Анализ масштабируемости

Размер стенда (IoT-device/Edge)	Благосостояние (%)	Эффективность (%)	Справедливость	Время (мс)
Малый (10/20)	89.2	91.5	0.82	45
Средний (25/50)	91.8	88.7	0.84	78
Большой (50/100)	90.3	86.4	0.81	175

Выводы:

- Производительность остается стабильной при увеличении размера системы
- Время конвергенции растет линейно
- Справедливость распределения практически не изменяется

Результаты сравнительного анализа

Алгоритм	Эффективность распределения	Справедливость (Джайн)	Время конвергенции
Наш метод (VCG)	85-92%	0.78-0.85	50-100 мс
MADRL (DDPG)	80-88%	0.70-0.80	200-500 мс
Contract Net Protocol	65-70%	0.72-0.82	180-200 мс
Random Allocation	45-55%	0.60-0.70	< 10 мс
FCFS	55-65%	0.55-0.65	< 20 мс
Proportional Fair	70-78%	0.85-0.90	100-200 мс
Game Theory	75-82%	0.72-0.82	150-300 мс
Bio-inspired (PSO)	78-85%	0.75-0.82	300-800 мс

Базовые алгоритмы из литературы:

- **MADRL**: Многоагентное обучение с подкреплением [X. Wang et al.]
- **Contract Net**: Классический протокол согласования в многоагентных системах [H. Hao et al.]
- **First-Come-First-Served (FCFS):** Обслуживание в порядке поступления запросов [T. Ahanger et al.]
- **Random Allocation**: Случайное распределение [T. Ahanger et al.]
- **Proportional Fair**: Пропорционально справедливое распределение [C. Zhang et al.]
- **Game Theory**: Алгоритмы на основе равновесия Нэша [Y.Huang et al.]
- **PSO**: Биоинспирированные алгоритмы оптимизации роя частиц [S. Kuma et al.]

Результаты альтернативных алгоритмов были **нормализованы и экстраполированы** на параметры нашего стенда с учетом:

- Масштабирования производительности по числу узлов
- Адаптации к нашей топологии сети
- Приведения метрик к единому базису измерения

Заключение

Разработано решение*, которое включает:

- Децентрализованный аукционный протокол Учет топологии и задержек
- Математические гарантии оптимальности и справедливости
- Устойчивость к масштабированию системы

Перспективы развития

- Гибридные аукционные схемы, сочетающие различные типы аукционов
- Расширение параметров системы (например, из метрик QoS)
- Учет мобильности и динамики сети (изменение топологии сети)
- Реализация на реальном оборудовании

* GitHub: https://github.com/Dima12101/EdgeAgentAuction