Обзор лабораторной работы: Что это такое и зачем?

Эта лабораторная работа посвящена изучению **системы массового обслуживания (СМО) с приоритетным обслуживанием**. Здесь заявки (пакеты) имеют разные приоритеты (высокий, средний, низкий), и канал обслуживает сначала более приоритетные, даже если низкоприоритетные ждут дольше. Это модель для сетей, где важные пакеты (например, голос) должны обрабатываться быстрее данных. Нет потерь (бесконечная очередь), но низкоприоритетные могут ждать долго. Ключевой показатель — **время доставки T** (ожидание W + обслуживание t=1), которое отличается по приоритетам и зависит от нагрузки ρ = a (суммарная интенсивность).

Цель: Освоить AnyLogic, построить модель M/M/1 с приоритетами (3 уровня), сравнить симуляцию с теорией, модифицировать в M/D/1 (детерминированное обслуживание) и изучить влияние на T.

План работы:

1. Построить и валидировать модель M/M/1 с приоритетами.
2. Сравнить имитацию и аналитику для M/M/1.
3. Модифицировать в G/G/1 (но в методичке M/D/1 как пример G) и исследовать.
4. Выводы.

Чтобы защититься: Расскажи по шагам, покажи модель в AnyLogic, таблицы, графики T от a для разных приоритетов. Объясни формулы, почему T разное (приоритеты влияют на ожидание). Я разобью просто, с советами.

1. Что такое СМО с приоритетным обслуживанием? (Теоретическая основа)

- **Заявки**: 3 потока с приоритетами (2 — высший, 1 — средний, 0 — низший). Приходят по пуассоновскому (M), интенсивность a_i для каждого (обычно равные, sum a_i = a).
- **Очередь**: Бесконечная, но приоритетная (Priority Queue): сортировка по prio (высший prio — вперёд). Дисциплина: Non-preemptive (невытесняющая) — текущую заявку не прерывают, но следующую берут по приоритету.
- **Обслуживание**: Один канал (1), время t — экспоненциальное (M, среднее 1) или детерминированное (D=1).
- **Правила**: Заявка приходит → в очередь с prio. Канал берёт из очереди самую приоритетную. Нет потерь.
- **Нагрузка ρ = a <1** (для стабильности). ρ_i = a_i.

- **Ключевой показатель**: T_k = W_k + t, где W_k — ожидание для приоритета k (k=1 высший, k=3 низший в формулах).

Это модель для QoS в сетях (quality of service). Симулируешь, чтобы увидеть, как приоритеты снижают T для важных заявок, но увеличивают для низких.

2. Построение модели в AnyLogic (Шаг 2.1–2.2)

- **Элементы модели** (Process Modeling Library, из прошлых лаб):
  - **3 Source (source0, source1, source2)**: Для каждого приоритета. Arrivals: exponential(1/a) (интервалы). New agent: Packet с параметрами origTime = time() (время создания), prio = 0/1/2 (низ/сред/выс).
  - **Queue**: Одна очередь. Capacity: Infinite. Type: Agent (Packet). Queue discipline: Priority-based, comparator: entity.prio (высший prio — max, так что prio=2 вперёд). Enable preemption: False (невытесняющая).
  - **Delay**: Обслуживание. Delay time: exponential(1) для M/M. Capacity: 1. On enter: ничего; On exit: data.add(time() - agent.origTime) для общей гистограммы T; data0/1/2.add по prio.
  - **Sink**: Конец. On enter: Обнови переменные T если нужно.
  - **Histogram Data (data, data0, data1, data2)**: Для сбора T общих и по prio (add в delay On exit: if(agent.prio==0) data0.add(...)).
  - **Histogram**: Визуализация data (bins=30, range 0-50).
  - **Параметр a**: double, нагрузка на один source (total a*3).
  - **Slider**: Для a или prio.
- **Структура**: source2 (prio2) → queue; source1 → queue; source0 → queue; затем queue → delay → sink (рис.1, 3 источника в одну очередь).

Настройки по рисункам 2–5:

- Source1 (пример для prio1): exponential(1/a), agent.prio=1.
- Queue: Priority-based, comparator: -compareTo(entity1.prio, entity2.prio) или entity.prio descending (высший prio max).
- Delay: exponential(1).
- Sink: Нет специфики.

3. Валидация модели (Шаг 2.3)

- **Проверка работы (2.3.1)**: Запусти с a=0.9/3 ≈0.3 на source (total ρ=0.9). Max скорость. Среднее T≈5; для prio2≈2.1, prio1≈3.3, prio0≈9.5 (рис.6). Если ошибки — проверь prio в агентах и comparator в queue.
- **Сравнение с аналитикой (2.3.2)**:

- **Формулы для M/M/1 с приоритетами** (невытесняющие):
  - Для низших приоритетов (K=2,3): W_k = [R_k / (1 - sum_{i=1}^{k-1} ρ_i)] * (1 - sum_{i=1}^k ρ_i)^{-1}, где R_k = sum_{i=1}^k ρ_i * (σ_i^2 + t_i^2)/2, но для M σ= t=1, упрощается до given формул.
  - Для высшего (K=1): W_1 = sum_{i=1}^M ρ_i / (2*(1 - sum_{i=1}^M ρ_i)) или как в методичке: W_1 = sum ρ_i * t / (1 - ρ_total).
  - T_k = W_k + t=1.
  - Примечание: K=1 — высший (prio2), K=3 — низший (prio0). ρ_i равны (a/3 каждая).
- Запусти для a=0.1..0.99 (total ρ=a), заполни таблицу 1: Для каждого a — T для prio0/1/2 из симуляции (ИМ), и аналитика для K=3/2/1.
- Графики: T от a для каждого prio (ИМ). Ожидай: T растёт с a, но для высшего prio T ниже (ближе к 1/(1-ρ_high)), для низшего — сильно растёт.

Результаты ИМ близки к АМ (разница от случайности, >10^5 заявок).

4. Исследование СМО M/D/1 с приоритетным обслуживанием (Шаг 3)

- **Модификация**: Измени delay: Delay time = 1 (constant) для D (детерминированное).
- **Эксперименты**: Повтори как в 2.3.2 для a=0.1..0.99.
  - **Формулы для M/D/1 приоритетами**: Аналог, но с σ_i=0 для D (variance=0). W_k = [sum_{i=1}^M (ρ_i/2 * (σ_i^2/t_i + t_i)) / (1 - sum_{i=1}^{k-1} ρ_i)] / (1 - sum_{i=1}^k ρ_i), упрощённо как в методичке.
  - Заполни таблицу 2: T для prio0/1/2 ИМ и АМ (K=3/2/1).
  - Графики: T от a для каждого prio (ИМ).
- Что увидишь: В M/D T ниже, чем в M/M, т.к. постоянное обслуживание (σ=0) снижает вариабельность, очереди короче. Разница заметна при высокой a: низкий prio выигрывает больше.

5. Выводы (Ответы на вопросы — готовься сказать преподавателю)

1. **По построению имитационной модели СМО с приоритетным обслуживанием**: Модель с 3 source (по prio), одной priority queue (comparator по entity.prio), delay, sink. Добавлены histogram data для T по prio. Легко из прошлых лаб, валидация показывает правильную дифференциацию T (высший prio — минимальное ожидание).
2. **По имитации и аналитике M/M/1 с приоритетами**: ИМ близко к АМ (разница <0.1 при больших runs). T растёт с a, но приоритеты "защищают" высшие заявки (T≈1/(1-ρ_high)), низшие страдают (T>>1/(1-ρ)). Подтверждает теорию: при ρ=0.9 высший T~2, низший ~10.
3. **По исследованию M/D/1 с приоритетами**: В детерминированном обслуживании T ниже на 20–50% vs M/M, особенно для низких prio, т.к. нет длинных "хвостов" в времени

обслуживания. AM работает (с σ=0), симуляция показывает преимущество D для стабильности. Полезно для сетей с фиксированными пакетами.