МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»

Методичні вказівки і завдання

до виконання лабораторних робіт
з курсу «Програмування розподілених систем
обробки даних»

2011

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

Методичні вказівки і завдання

до виконання лабораторних робіт
з курсу «Програмування розподілених систем
обробки даних»
(для студентів спеціальності 6.080403 „Програмне забезпечення автоматизованих систем")

Укладачі:
А. М. Гізатулін, к.е.н., доц.
Ю. В. Попов, к.т.н., доц.

Розглянуто на засіданні кафедри
прикладної математики і інформатики
Протокол № 2 від __.__.2011

Затверджено на засіданні
Навчально-видавничої ради ДонНТУ
Протокол № ___ від ___.___.2011

2011

УДК 004.052

Методичні вказівки і завдання до виконання лабораторних робіт з курсу «Програмування розподілених систем обробки даних» / для студентів спеціальності 6.080403 Програмне забезпечення автоматизованих систем / Укладачі Гізатулін А.М., Попов Ю.В.– Донецьк: ДонНТУ, 2011. – 26 с.

Методичні вказівки і завдання до виконання лабораторних робіт з курсу «Програмування розподілених систем обробки даних» підготовлені на основі типової програми курсу і направлені на вивчення методології, методики та інструментарію програмування розподілених систем обробки даних, їх аналізу та використання. Метою лабораторного практикуму є формування системи практичних знань у галузі дослідження та програмування розподілених систем обробки даних.

Укладачі: А.М. Гізатулін, к.е.н., доц.
Ю.В. Попов, к.т.н., доц.

СОДЕРЖАНИЕ

Лабораторная работа №1 Разработка виртуальных часов …… 5
Лабораторная работа №2 Разработка списка событий ……. 7
Лабораторная работа №3 Структурно-функциональная модель схемы ……. 10
Лабораторная работа №4 Последовательный координатор процесса моделирования …….. 14
Лабораторная работа №5 Консервативный коммуникационный интерфейс . 18
Лабораторная работа №6 Система коммуникаций ……... 21
Лабораторная работа №7 Консервативный координатор процесса моделирования …….. 23
Список рекомендуемой литературы ……... 26

Лабораторная работа №1

Тема: **Разработка виртуальных часов**

Цель работы

Изучить способы управления виртуальным временем в системах логического моделирования.

Ход работы

Моделирование выполняется в дискретном виртуальном времени VT. Один тик VT соответствует одному такту работы схемы. Моделирование начинается в момент VT=0 и заканчивается в конечный момент (указывается пользователем), либо когда больше нет событий. Каждому событию соответствует момент VT, когда это событие наступит.

При разработке виртуальных часов следует учитывать:

- виртуальные часы показывают виртуальное время;
- Время – это положительное целое число;
- В начальный момент времени часы показывают 0;
- Часы переводятся вперёд, когда в списке событий больше нет событий, запланированных на текущий момент времени;
- В некоторых алгоритмах часы могут переводиться назад;
- Нужно предусмотреть средства синхронизации, чтобы разные потоки могли считывать показания часов.

Поля виртуальных часов:

- Текущее время (целое число)
- Могут ли часы переводиться назад (флаг)

Методы виртуальных часов

- Установить текущее время (метод должен проверять, что время положительное; назад можно переводить только если установлен флаг)
- Узнать текущее время
- Сбросить текущее время в ноль
- Установить флаг перевода назад
- Получить флаг перевода назад

Задание

1. Создать класс виртуальных часов.
2. Создать Unit-тесты для всех методов виртуальных часов.
3. Создать основной класс, в котором будут вызываться все Unit-тесты.

Содержание отчета

1. Титульный лист.
2. Тема лабораторной работы.
3. Задание на лабораторную работу.
4. Распечатка исходного кода класса виртуальных часов.
5. Распечатка исходного кода всех unit-тестов
6. Распечатка исходного главного класса
7. Выводы.

Контрольные вопросы

1. Какие методы должны реализовывать виртуальные часы?
2. Какие поля есть у виртуальных часов?
3. В чём измеряется виртуальное время?
4. Может ли виртуальное время уменьшаться?

Лабораторная работа №2

Тема: **Разработка списка событий**

Цель работы
Изучить способы управления потоком событий в системах логического моделирования.

Ход работы

Список событий является хранилищем событий в пределах МодПр. Список событий сортирует события по порядку их наступления в виртуальном времени в пределах МодПр. В начале моделирования список событий содержит все события входного воздействия. События всегда выбираются только в порядке увеличения виртуального времени их наступления. Добавляться события в список могут в случайном порядке. Список событий изменяется дважды при обработке каждого события - первый раз при генерации нового события, второй раз - в момент обработки события, при удалении. В начале цикла моделирования события, запланированные на текущий момент виртуального времени выбираются из списка событий по одному. После применения всех таких событий производится вычисление новых значений сигналов на выходах элементов. Этим новым сигналам соответствуют события в будущем, которые записываются в список событий. После этого цикл моделирования повторяется. В консервативных и комбинированных протоколах список событий умеет ждать и возвращать управление в вызывающую процедуру только появления нового подходящего события. Список событий должен предусматривать средства синхронизации потоков.

Поля списка событий:

- Собственно список. Это массив структур, каждая из которых содержит три поля:
- Виртуальное время (целое)
- Номер узла (индекс в массиве узлов)
- Новое значение сигнала
- Время завершения моделирования
- LVTH (в оптимистических содержит +бесконечность)

Методы списка событий

- Получить время, номер узла и узел следующего события. Возвращает данные для события с самым маленьким виртуальным временем наступления. Это могут быть три независимых метода
- Добавить событие. Новые события добавляются так, чтобы не нарушить порядок сортировки списка
- Очистить. Вызывается в начале моделирования, чтобы очистить список событий. Метод должен сбрасывать в начальное состояние все поля объекта, в том числе и время завершения моделирования и LVTH
- Удалить следующее событие. Удаляет из списка следующее (с наименьшим временем наступления) событие
- Получить количество событий в списке. Возвращает общее количество событий в списке
- Получить/установить время окончания моделирования. При попытке добавить событие после времени окончания моделирования список событий ведёт себя так, как будто событие успешно добавлено, но фактически событие не добавляется в список
- Загрузить входное воздействие. Метод позволяет загрузить содержимое списка событий из файла
- Установить/получить LVTH. По умолчанию поле LVTH равно бесконечности, что соответствует оптимистической и должно быть так в последовательной модели модели. Во время консервативного моделирования LVTH меняется в процессе моделирования, отражая текущее предсказание времени.
- Ожидать подходящего события. Если время наступления следующего события меньше, чем LVTH, то этот метод ничего не делает. Иначе метод ждёт появления такого события. Подходящее событие может появиться в двух случаях: 1) Такое событие придёт от удалённого МодПр или 2) Увеличится LVTH. В Java есть специальные средства, позволяющие заблокировать поток до момента наступления некоторого события.

Во всех методах списка событий обязательно предусмотреть средства синхронизации потоков! Один (основной) поток получает события от удалённых МодПр по сети, а другой (вычислительный) – выполняет обработку событий. Вычислительный поток блокируется, если нет событий для обработки

Список событий должен быть реализован универсально, не зависимо от используемого протокола синхронизации (последовательный, консервативный, оптимистический, комбинированный).

Для проверки состояния полей допускается добавить специальные protected-методы. Класс-тест может наследовать основной класс – чтобы видеть protected-методы.

Список событий изменяется очень часто во время моделирования (дважды для каждого события). Наиболее ресурсоёмкие операции:

- найти место в списке, куда нужно добавить новое событие;
- добавить событие в список;
- удалить событие из списка.

Удобно использовать динамические списки.

Задание

1. Создать класс списка событий.
2. Создать Unit-тесты для всех методов списка событий.
3. Создать основной класс, в котором будут вызываться все Unit-тесты.

Содержание отчета

1. Титульный лист.
2. Тема лабораторной работы.
3. Задание на лабораторную работу.
4. Распечатка исходного кода класса списка событий.
5. Распечатка исходного кода всех unit-тестов
6. Распечатка исходного главного класса
7. Выводы.

Контрольные вопросы

1. Какие методы должен реализовывать список событий?
2. Какие поля есть у списка событий?
3. Что такое событие?
4. Как упорядочены события внутри списка событий?
5. При каких условиях список событий разблокируется?

Лабораторная работа №3

Тема: **Структурно-функциональная модель схемы**

Цель работы
Изучить способы представления структуры и функций логических схем в компьютерных программах.

Ход работы

Назначение СФМС:
- Хранение структуры моделируемой схемы: какие элементы, как между собой связаны
- Хранение состояния моделируемой схемы: значения сигналов во всех узлах
- Представление функций моделируемой схемы: уметь определять сигналы на выходах элементов по известным значениям сигналов на входах
- Ведение списка изменённых элементов - чтобы потом только на выходах этих элементов производить пересчёт сигналов
- Генерация новых событий в соответствии с изменяемыми сигналами на выходах элементов
- Представление внешних узлов, позволяющее передавать сигналы между узлами на разных МодПр (узлы имеют имя)
- Генерация внешних событий (в случае ленивой отмены)
- Загрузка схемы из файла перед началом моделирования
- Сохранение текущего состояния схемы в стеке состояний

Поля СФМС:
- Массив элементов. О каждом элементе известно:
  - функция элемента (И, ИЛИ, НЕ и т.д.)
  - имя элемента (в целях отладки)
  - список номеров узлов, сигналы с которых подаются на входы элемента (узлов-входов). Индекс в массиве узлов.
  - список номеров узлов-выходов
- Массив узлов. О каждом узле известно:
  - список элементов, на входы которых влияет этот узел. Используется для того, чтобы определить список изменённых элементов, на выходах которых следует пересчитать сигналы
  - имя узла (в целях отладки)
  - значение сигнала (истина, ложь, неопределённое состояния, возрастающий фронт, ниспадающий фронт и т.п.)
- Массив внешних входов. О каждом входе известно:
  - имя входа. По этому имени МодПр находит соответствие между локальным и удалённым внешним узлом - эти узлы должны иметь одинаковое имя.
  - номер локального узла (индекс в массиве узлов).
- Массив внешних выходов. О каждом внешнем выходе известно:
  - имя внешнего узла. Отправляется вместе с внешним событием, чтобы удалённый МодПр мог понять, на каком именно узле произошло событие;
  - номер локального узла (индекс в массиве узлов)
  - имя процессора - получателя сообщений о событиях на этом узле.0
- Список изменённых элементов. При изменении сигналов в узлах схемы сюда заносятся те элементы, входы которых присоединены к узлу с изменённым значением сигналов. Каждый отдельный элемент должен быть записан здесь ровно один раз, не зависимо от того, на скольких входах изменились значения сигналов.

Свойства СФМС:
- Посчитать количество элементов
- Посчитать количество узлов
- Определить количество входных узлов у элемента по его номеру
- Определить количество выходных узлов у элемента по его номеру
- Определить имя элемента по номеру элемента
- Определить количество внешних входов
- Определить количество внешних выходов
- Определить имя внешнего входа по номеру внешнего входа (индексу в массиве входов)
- Определить имя внешнего выхода по номеру внешнего выхода (индексу в массиве выходов)
- Определить имя узла по номеру узла (индексу в массиве узлов)

- Определить значение сигнала в узле по номеру узла
- Определить имя функции элемента
- Определить, была ли успешно загружена схема

Методы СФМС:

- Сброс в начальное состояние всех полей СФМС
- Загрузить схему из файла
- Получить имя внешнего выхода и имя удалённого МодПр по номеру элемента и номере выхода элемента. Используется при возникновении внешнего события, чтобы узнать куда нужно отправить сообщение о событии. Допустимо для ускорения хранить эту информацию в готовом виде в объекте элемента
- Определить значение сигнала на входе элемента. По номеру элемента (индексу в массиве элементов) и номеру входа элемента определяет номер узла (индекс в массиве узлов) и затем возвращает значение сигнала в этом узле
- Определить номер узла (индекс в массиве узлов) по имени внешнего входного узла
- Установить значение сигнала в узле схемы. Может дополнительно добавлять элементы в список изменённых элементов. Эта возможность определяется дополнительным третьим аргументом метода - например, при откате такая возможность не требуется.
- Сбросить список изменённых элементов
- Посчитать значения сигналов на выходах элементов. При этом важно, что за один вызов метода значения на выходах одного элемента должны быть обновлены не более одного раза. Обычно это так, если в списке изменённых элементов нет повторов. Метод принимает на вход текущее LVT и генерирует новые события, с учётом текущего времени и задержки на элементах. Не нужно генерировать событие, если значение на выходне изменилось. После завершения метода список изменённых элементов должен быть обнулён.

Удобно сделать базовый класс "Элемент" с абстрактным методом для вычисления значений сигналов на выходах и множество наследников по каждому виду элементов (И, ИЛИ, НЕ и т.д.)

Список изменённых элементов имеет смысл делать отсортированным - чтобы быстрее определять, добавлен уже такой элемент или нет

Удобно заводить методы для каждого отдельного действия. Например, "добавить элемент в список изменённых элементов" - отдельный метод;

В последовательном и консервативном протоколе синхронизации СФМС используется только вычислительным потоком

Имеет смысл предусмотреть средства синхронизации потоков, так как в оптимистических протоколах откат часто выполняется управляющим потоком (при откате меняются значения сигналов в узлах схемы)

Удобно нигде, кроме специализированных методов-свойств не получать прямой доступ к полям.

Задание

1. Создать класс структурно-функциональной модели схемы.
2. Создать Unit-тесты для всех методов структурно-функциональной модели схемы.
3. Создать основной класс, в котором будут вызываться все Unit-тесты.

Содержание отчета

1. Титульный лист.
2. Тема лабораторной работы.
3. Задание на лабораторную работу.
4. Распечатка исходного кода класса структурно-функциональной модели схемы.
5. Распечатка исходного кода всех unit-тестов
6. Распечатка исходного главного класса
7. Выводы.

Контрольные вопросы

1. Какие методы должны быть в СФСМ?
2. Какие поля есть в СФСМ?
3. Какие свойства есть в СФСМ
4. Для чего используется СФСМ?
5. Что произойдёт, если в один момент виртуального времени на элементы будут пересчитаны значения выходных сигналов дважды?

Лабораторная работа №4

Тема: **Последовательный координатор процесса моделирования**

Цель работы

Изучить алгоритмы последовательного моделирования цифровых логических систем.

Ход работы

Процессор, который обрабатывает все события в моделируемой системе события по очереди, одно за другим, называется последовательным моделирующим процессором. Архитектура последовательного моделирующего процессора представлена на рис. 4.1.

Последовательный моделирующий процессор содержит:

- список событий EVL. В этом списке содержатся все события, которые возникнут в моделируемой схеме. События в этом списке упорядочены в порядке увеличения виртуального времени их наступления;
- виртуальные часы VT. Показывают текущее виртуальное время;
- структурно-функциональная модель схемы СФМС. Содержит в себе структуру моделируемой схемы, описание функций используемых в схеме элементов и значения сигналов во всех узлах схемы;
- координатор процесса моделирования КПМ. Управляет порядком работы всех объектов моделирующего процессора.

КПМ выбирает из списка событий события, запланированные на текущий момент виртуального времени (рис. 4.2). Передает в СФМС данные для установки значений в узлах схемы в соответствие с выбираемыми событиями. Когда в списке событий больше нет событий, запланированных на текущий момент виртуального времени, КПМ передает СФМС команду пересчитать значения сигналов на выходах элементов в соответствие с измененными входными значениями сигналов. При этом могут возникнуть новые события, которые записываются в список событий.

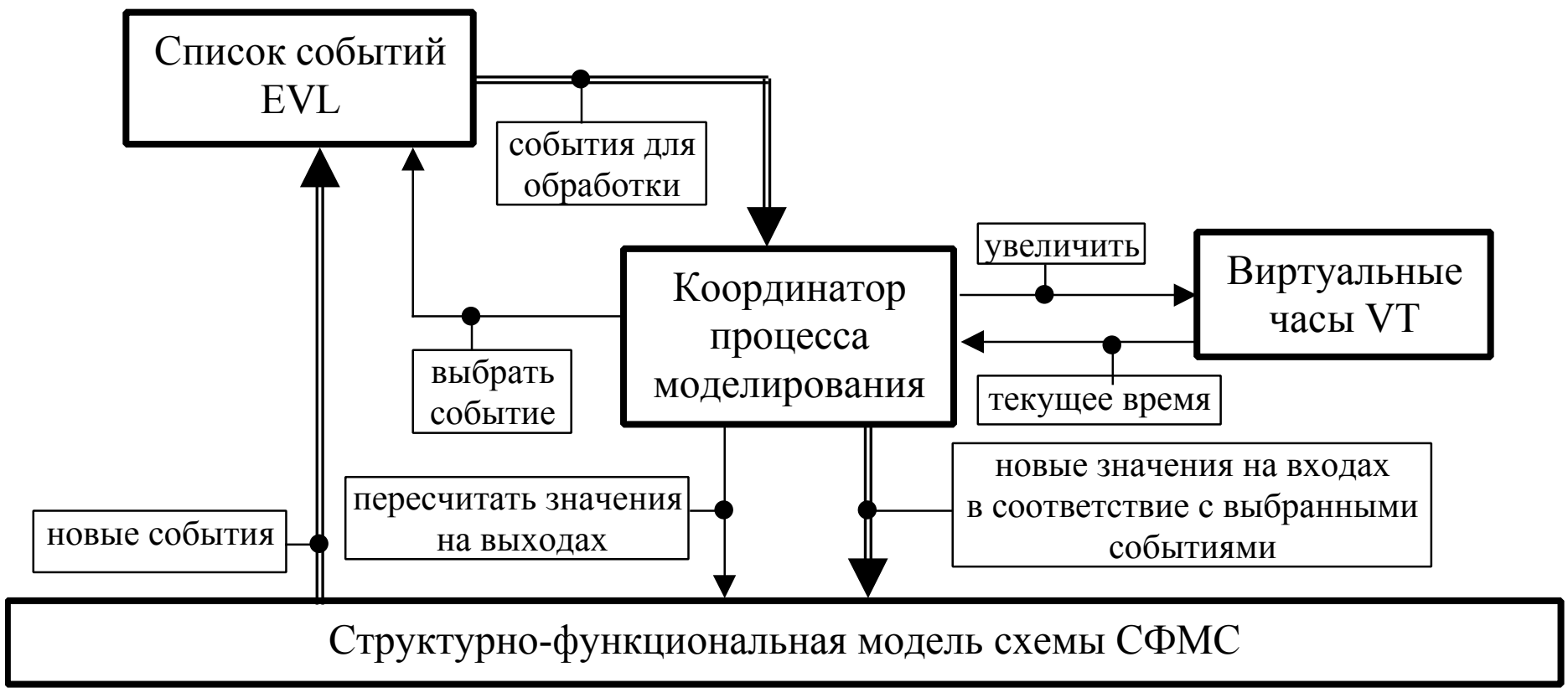

Рисунок 4.1 - Архитектура последовательного моделирующего процессора

```
пока LEVL.есть_события
begin
  LVT:=LEVL.время_следующего_события;
  пока LEVL.время_следующего_события=LVT
  begin
    узел:=LEVL.место_следующего_события;
    значение:=LEVL.значение_следующего_события;
    LEVL.удалить_следующее_событие();
    если СФМС.значение_в_узле(узел)=значение то
      //не изменилось
      перейти к началу цикла;
    СФМС.установить_значение_в_узле(узел,значение);
  end;
  СФМС.посчитать_значения_на_выходах_измененных_элементов();
end;
```

Рисунок 4.2 - Алгоритм работы последовательного координатора процесса моделирования

Время наступления этих событий определяется как текущее виртуальное время плюс неотрицательная задержка на соответствующем элементе, поэтому время наступления новых событий всегда больше текущего виртуального времени. Когда процесс пересчета завершен, КПМ увеличивает текущее виртуальное время до момента наступления следующего события в системе.

Задание

1. Разработать класс для последовательного координатора процесса моделирования.
2. Создать Unit-тесты для последовательного координатора процесса моделирования.
3. Создать класс, в котором будут вызываться все Unit-тесты.
4. Создать приложение для последовательного моделирования логической цифровой схемы.

Содержание отчета

1. Титульный лист.
2. Тема лабораторной работы.
3. Задание на лабораторную работу.
4. Распечатка исходного всех разработанных в работе классов.
5. Распечатка исходного кода всех unit-тестов
6. Распечатка исходного главного класса
7. Распечатка результатов моделирования
8. Выводы.

Контрольные вопросы

1. Какие объекты содержатся в последовательном МодПр?
2. Как события попадают в список событий?
3. Как в СФСМ появляется информация о новых значениях в узлах схемы?
4. Как КПМ узнаёт, когда нужно начать пересчитывать значения на выходах элементов?
5. Сколько раз за всё время моделирования выполнится внешний цикл КПМ?

Лабораторная работа №5

Тема: **Консервативный коммуникационный интерфейс**

Цель работы
Изучить методы синхронизации логических процессов в распределённых системах обработки данных.

Ход работы

Консервативный коммуникационный інтерфейс предназначен для:
- управления глобальным порядком обработки событий в системе;
- вычисления глобальных состояний;
- управления в режиме тупика (обнаружение, восстановление, предотвращение);
- получения и передачи сообщений и событиях;
- блокировки и разблокировки КПМ;
- вычисления LVTH.

Поля консервативного коммуникационного интерфейса:
- Входные буферы. Это массив, в котором для каждого входного узла записан локальный номер узла, на который подаются сигналы и время последнего пришедшего по этому узлу события (канальные часы).
- Выходные буферы. Это массив, в котором для каждого выходного узла записано значение и время последнего отправленного по этому узлу сообщения о событии.
- Горизонт расширения локального виртуального времени. Целое число, которое обновляется автоматически при получении очередного сообщения о событии
- Цвет МодПр - красный или белый. Используется для обнаружения состояния тупика при помощи маркера

Методы консервативного коммуникационного интерфейса:
- Получить сообщение. Универсальный метод, который проводит первичный анализ полученного сообщения и вызывает, в зависимости от типа сообщения, специализированный метод. Два аргумента: имя удалённого процессора, строка с блоком даннях.
- Получить сообщение о событии. Изменяет соответствующий событию входной буфер. Вычисляет новое значение LVTH. Заносит событие в LEVL. Регистрирует активность. Два аргумента: имя удалённого процессора, строка с блоком даннях.
- Получить сообщение о маркере. В этом методе реализуется основной алгоритм работы маркера. Два аргумента: имя удалённого процессора, строка с блоком даннях.
- Получить сообщение о необходимости продвижения LVT. Увеличивает LVTH вне зависимости от показаний канальных часов – используется при восстановлении системы из состояния тупика. При этом разблокируется LEVL. Два аргумента: имя удалённого процессора, строка с блоком даннях.
- Отправить сообщение. Универсальный метод, который вызывается специализированными методами для передачи сообщения по сети. На вход подаётся готовое сформированное сообщение. Два аргумента: имя удалённого процессора, строка с блоком данных.
- Отправить сообщение о событии. Изменяет выходной буфер. Проверяет, что сигнал изменился. Регистрирует активность. Формирует сообщение о событии. Три аргумента: имя узла, время, значение сигнала. Имя удалённого процессора определяется на основании имени узла.
- Отправить маркер. Формирует сообщение-маркер и передаёт его универсальному методу отправки сообщений. Три аргумента, которые соответствуют полям маркера. Имя удалённого процессора определяется на основании глобального порядка процессоров при помощи системы коммуникаций (берётся процессор, следующий после текущего).
- Отправить сообщение о необходимости увеличения LVT. Вызывается в основном алгоритме обработки маркера в случае обнаружения тупика. Два аргумента: имя удалённого процессора, LVTH.
- Закончить обработку событий в текущий момент времени. Вызывается из КПМ в момент, когда обработка событий в текущий момент времени завершена. Используется для рассылки пустых сообщений. Один аргумент – текущее время.
- Сброс. Сбрасывает значения всех полей в начальное состояние
- Получить LVTH. Возвращает значение поля LVTH.
- Регистрировать активность. Устанавливает цвет МодПр в белый.

Сообщения, передаваемые по сети:
- Сообщение о событии. Поля: глобальное имя узла, значение сигнала, время.

- Сообщение-маркер. Поля: время наступления следующего события в системе, процессор, на котором возникнет следующее событие, количество красных процессоров, которые посетил маркер.
- Увеличить LVTH

Сообщение удобно передавать в виде блока данных. Блок данных – это произвольная строка. Блок данных может быть как двоичным, так и текстовым (не имеет значения для системы коммуникаций). Метод отправки сообщения формирует блок данных, а метод получения сообщения разбирает полученный блок данных.

Пример текстовых блоков для разных сообщений
Сообщение о событии: event|n1@p1|1|129
“event” – признак того, что это сообщение о событии
Означает, что событие в момент времени 129 произойдёт на узле с именем «n1@p1», новое значение сигнала равно 1.

Маркер: marker|150|p3|2
“marker” – признак того, что это сообщение о маркере
Означает, что следующее событие в системе произойдёт на процессоре p3 в момент времени 150, и что было посещено 2 красных процессора.

Сообщение о необходимости увеличить LVTH: lvth|150
«lvth» - признак того, что это сообщение о необходимости увеличить LVTH без учёта значений канальных часов
Означает, что процессор-получатель сообщения должен увеличить своё значение LVTH до 150.

Задание
1. Разработать класс для консервативного коммуникационного интерфейса.
2. Создать Unit-тесты для всех методов консервативного коммуникационного интерфейса.
3. Создать класс, в котором будут вызываться все Unit-тесты.

Содержание отчета
1. Титульный лист.
2. Тема лабораторной работы.
3. Задание на лабораторную работу.
4. Распечатка исходного всех разработанных в работе классов.
5. Распечатка исходного кода всех unit-тестов
6. Распечатка исходного главного класса
7. Выводы.

Контрольные вопросы
1. Какие методы реализуются консервативным коммуникационным интерфейсом?
2. Какие поля есть в консервативном КИ?
3. Для чего предназначен КИ?
4. В чём отличие консервативного КИ от оптимистического?
5. Как КИ узнаёт о событиях на разрезанных узлах схемы?

Лабораторная работа №6

Тема: **Система коммуникаций**

Цель работы

Изучить методы взаимодействия логических процессов в распределённых системах обработки данных.

Ход работы

Система коммуникаций предназначена для передача блоков данных между процессорами, установки сеансов связи и идентификации процессоров по имени. Система коммуникаций является интерфейсом между коммуникационным интерфейсом и средствами связи, предоставляемыми Java.

Этот модуль – универсальный для консервативного, оптимистического и комбинированного протоколов синхронизации.

Поля системы коммуникаций

- Список удалённых МодПр. О каждом МодПр известно: имя МодПр (строка, уникальная), IP-адрес, номер порта, клиентский сокет.
- Локальное имя МодПр
- Серверный сокет

Методы системы коммуникаций

- Отправить сообщение. Два аргумента: имя МодПр, блок данных.
- Получить сообщение. Вызывает метод получения сообщения КИ, при этом ему передаётся имя удалённого МодПр – отправителя сообщения и полученный блок данных.
- Инициализировать серверный сокет. Вызывается один раз при запуске программы. Номер порта задаётся в настройках..
- Инициализировать клиентские сокеты. Загружает из настроек список удалённых МодПр и устанавливает с ними соединение. При этом связь со старыми МодПр закрывается.
- Сброс. Сбрасывает в начальное состояние все поля (список удалённых МодПр – пустой, локальное имя МодПр не задано)
- Получить имя следующего МодПр. Выстраивает все МодПр (удалённые и локальный) в один отсортированный список и возвращает тот МодПр, который следует непосредственно за текущим. Способ сортировки – произвольный, но должен приводить к одинаковому отсортированному списку на всех МодПр. Можно сортировать по имени МодПр или по связке (IP-адрес, порт)

В реальной системе может потребоваться возможность одновременного моделирования нескольких независимых задач моделирования. В таком случае в систему должен быть введён сервер, который управлял бы тем, какие процессоры какие задачи будут решать (рис. 6.1).

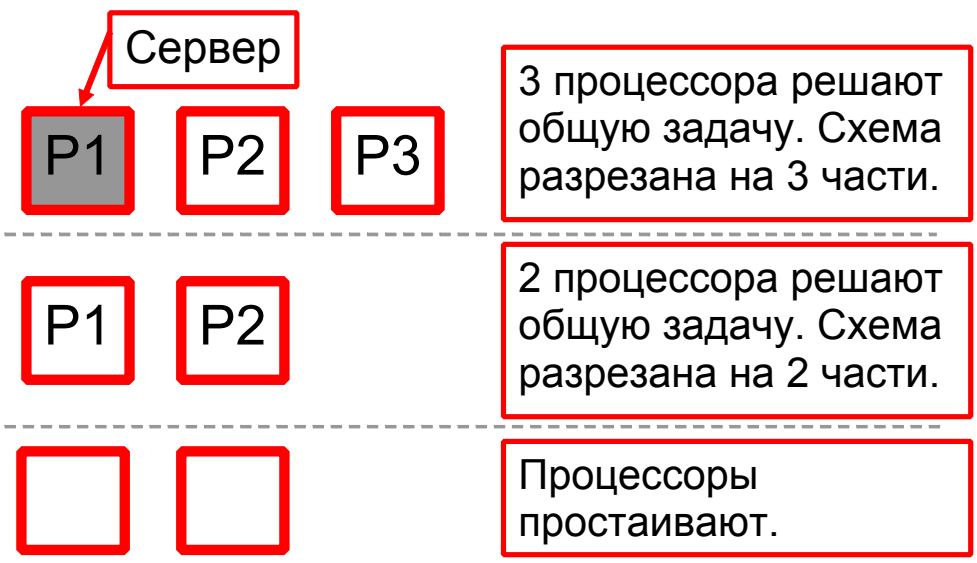

Рисунок 6.1 – Несколько задач моделирования решаются параллельно. У процессоров, решающих резные задачи, могут быть одинаковые имена.

Задание

1. Разработать класс для системы коммуникаций.
2. Создать Unit-тесты для всех методов системы коммуникаций.
3. Создать класс, в котором будут вызываться все Unit-тесты.

Содержание отчета

1. Титульный лист.
2. Тема лабораторной работы.
3. Задание на лабораторную работу.
4. Распечатка исходного всех разработанных в работе классов.
5. Распечатка исходного кода всех unit-тестов
6. Распечатка исходного главного класса
7. Выводы.

Контрольные вопросы

1. Какие методы реализуются системой коммуникаций?
2. Какие поля есть в системе коммуникаций?
3. Для чего предназначена система коммуникаций?
4. Как система коммуникаций позволяет определить порядок передачи маркера между моделирующими процессорами?
5. Как можно уменьшить количество открытых соединений в системе коммуникаций?

Лабораторная работа №7

Тема: **Консервативный координатор процесса моделирования**

Цель работы

Изучить алгоритм распределённого логического моделирования цифровых устройств.

Ход работы

Консервативный коммуникационный интерфейс содержит один входной буфер IBi и каналные часы CCi для каждого входного канала связи. IB временно сохраняет входящие сообщения в порядке FIFO. CCi сохраняют копию времени последнего пришедшего по этому каналу связи сообщения. В начале CCi устанавливается в ноль. LVTH=min(CCi) – горизонт, до которого можно расширять LVT, моделируя внутренние и внешние события. Не могут появиться события со временем, меньшим чем LVTH. CI даёт команду КПМ обрабатывать внутренние и внешние события до тех пор, пока LVT<=LVTH. КПМ обрабатытвает события так, как если бы это была последовательная система моделирования, но ждёт, если время следующего события больше, чем LVTH (рис. 7.1).

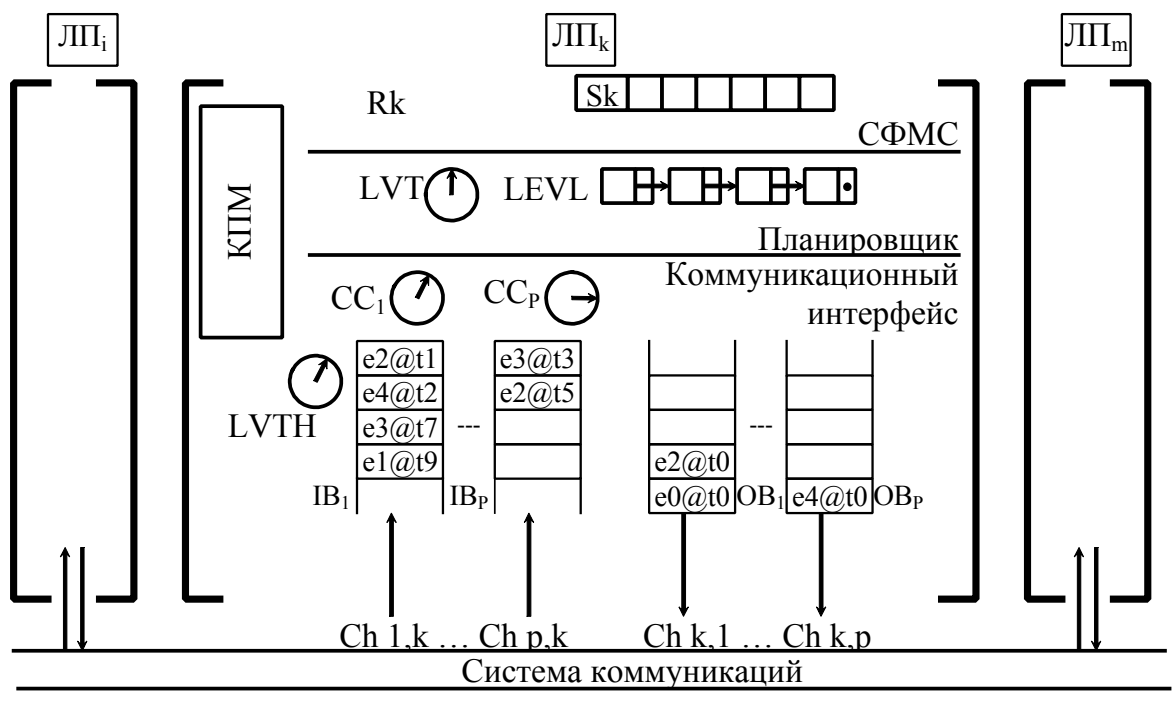

Рисунок 7.1 – Архитектура консервативного логического процесса.

При обработке событий КПМ может сгенерировать события в будущем для удалённых ЛП. Для каждого из таких событий, конструируется сообщение и помещяется в выходной FIFO-буфер OB. Сообщения выбираются из OB системой коммуникаций и доставляются удалённым процессорам. Алгоритм работы консервативного координатора процесса моделирования приведён на рис. 7.2.

```
while LEVL.has_events() or
      CI.LVTH<? do // могут возникнуть внешние соб.
begin
  if LEVL.next_event_ts()>CI.LVTH then
  begin
    wait_until_an_event_message_arrives();
    continue;
  end;
  LVT:=LEVL.next_event_ts();
  while LEVL.next_event_ts()==LVT do
  begin
    node:=LEVL.next_event_node();
    value:=LEVL.next_event_value();
    LEVL.rermove_next_event();
    if value==NO_CHANGE then //ничего не поменялось
      continue;
    if SFSM.node_value(node)==value then
      continue; // value is not changed
    SFSM.set_node_value(node,value);
  end;
  SFSM.update_affected_elements();
  CI.send_external_events();
```

```
end;
```

Рисунок 7.2 – Алгоритм работы консервативного координатора процесса моделирования.

Задание

1. Разработать класс для консервативного координатора процесса моделирования.
2. Предусмотреть средства для борьбы с тупиками.
3. Разработать приложение для распределённого логического моделирования цифровых устройств.

Содержание отчета

1. Титульный лист.
2. Тема лабораторной работы.
3. Задание на лабораторную работу.
4. Распечатка исходного всех разработанных в работе классов.
5. Распечатка результатов моделирования
6. Экранные формы
7. Выводы.

Контрольные вопросы

1. Что делает координатор процесса моделирования?
2. В чём отличие консервативного КПМ от оптимистического?
3. Что произойдёт, если событие от удалённого МодПр будет добавлено в локальный список событий в момент, когда КПМ выбирает и применяет к СФСМ события, запланированные на текущий момент виртуального времени?
4. Что произойдёт, если придёт запоздавшее событие?
5. При каких условиях КПМ может начать ждать?
6. При каких условиях КПМ перестаёт ждать?

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Kshemkalyani A.D., Singhal M. Distributed Computing: Principles, Algorithms, and Systems. - London: Cambridge university press, 2008. – 736 p.

2. Tools and Environments for Parallel and Distributed Systems (International Series in Software Engineering) /A. Zaky, T. Lewis. – London: Springer, 1996. - 320 p.

3. G. Coulouris, J. Dollimore, T. Kindberg, G. Blair. Distributed Systems: Concepts and Design. – London: Addison Wesley, 2011. - 1008 p.

4. Formal Methods for Distributed System Development (IFIP Advances in Information and Communication Technology) / Tommaso Bolognesi, Diego Latella. – London: Springer, 2000. - 424 p.

5. Fleischmann A., Tischer J. Distributed Systems: Software Design and Implementation. – London: Springer, 1994. – 390 p.

6. Bal H. E. Programming Distributed Systems. – NY: Silicon Pr, 1990. – 282 p.

7. Pattison T. Programming Distributed Applications with COM+ and Microsoft Visual Basic (DV-MPS Programming). – NY: Microsoft Press, 2000. – 456 p.