БГУИР

Кафедра ЭВМ

Лабораторная работа №4

Случайные методы доступа к моноканалу

Выполнил Проверил:

Студент группы 750502 Глоба А.А

Никанов И.В.

Минск 2019

**Основные теоретические сведения:**

Различные алгоритмы доступа к моноканалу разрабатывают по причине необходимости разрешения конфликтов между станциями при взаимодействии посредством разделяемой СрПД.

В первую очередь затрагиваются передатчики, то есть активные компоненты системы. Проблема заключается в «столкновениях» конкурирующих передатчиков. Пассивные по своей природе приемники априори конфликтовать не могут. Хотя количество приемников всегда ограничивается, так как передатчики имеют конечную нагрузочную способность.Если находящиеся в равных условиях два либо более передатчиков одновременно выдают сигналы в СрПД (например, устанавливают соответствующие уровни напряжения), то возникает противоречие.

Таковое единовременно неразрешимое противоречие принято называть

*коллизией* (collision).

Коллизия может быть как логической (информационный конфликт) так и физической (несовместимые физические процессы). Обычно коллизия возникает при попытках установить противоположные логические уровни. Кроме всего прочего, физическая коллизия чревата выходом из строя передатчиков, даже при попытках установить одинаковые логические уровни, так как многие среды не допускают наличие более чем одного активного усилителя сигнала без применения специальных схемотехнических решений. Классическим способом защиты оборудования от коллизий является так называемая гальваническая развязка (трансформаторная либо оптронная). При попытках установить разные уровни, как правило, наблюдаются эффекты «зануления» и «заединичивания» -- в зависимости от особенностей элементной базы.

Ситуация с коллизией может затрагивать только станции, подключенные к одной СрПД, то есть сегмент компьютерной сети. Сегмент, в котором возможно возникновение коллизий называется *доменом коллизий* (collision domain).

Понятие коллизии относится не только к сигналу, а и к пакету.

Существуют два основных подхода к проблеме коллизий:

1. Не допускать коллизии вообще, то есть использовать детерминированные методы доступа к моноканалу.

2. Допускать коллизии и каким-то образом выходить из них, что достижимо только использованием случайных методов доступа к моноканалу.

Во втором случае так же можно выделить два подхода:

1. Не обращать внимание на причины возникновения коллизий, а упор делать на способ выхода из них.

2. Пытаться предотвращать коллизии тем самым максимально снижая их количество, ну а если коллизии все-таки возникают, то «тяжело» выходить из них.

Таким образом, все методы доступа к моноканалу делят на:

1. Случайные (contention-based).

2. Детерминированные (controlled).

Все случайные методы основаны на использовании генератора случайных чисел (поэтому их так и называют), который позволяет делать случайные задержки при доступе к моноканалу, а значит и с определенной степенью вероятности избегать коллизии.

На эффективность случайных методов наиболее существенное влияние оказывают следующие факторы:

-- количество взаимодействующих станций;

-- инертность среды передачи данных;

-- длина кадра;

-- частота синхронизации.

С точки зрения изучения случайных методов доступа к моноканалу наиболее наглядным примером является классический алгоритм CSMA/CD (Carrier Sense Multiple Access with Collision Detection) -- множественный доступ с прослушиванием несущей и обнаружением коллизий, описанный в стандарте Ethernet (IEEE 802.3).

В данном алгоритме при обнаружении коллизии станциям передаётся jam-сигнал, который свидетельствует о том, что данное сообщение необходимо передать повторно. При этом каждая станция ожидает различное случайное время. Это сделано для того, чтобы в следующий раз при передаче вероятность их одновременной доступа к каналу уменьшилось. И если коллизия происходит повторно, то границы случайных чисел для времени ожидания увеличиваются.

Задержка перед началом очередной попытки передачи после коллизии (backoff) измеряется в такназываемых слот-таймах, количество которых является случайным целым числом r:

0 ≤ r ≤ 2k , где k = min (n, 10), где n -- номер попытки.

После превышения счетчиком попыток некоторого порогового значения дальнейшие попытки считаются бесперспективными. Значение k не может быть больше 10.

Качество диспетчеризации при обработке коллизий по большому счету зависит от одного базового параметра. Cлот-тайм (slot time) является минимальной неделимой единицей времени при диспетчеризации и подбирается с учетом многих других параметров. По крайней мере, он должен быть больше суммы удвоенного времени прохождения сигнала по сегменту и времени передачи jam-сигнала.

В стандарт заложен механизм ускорения распределенного обнаружения коллизий, заключающийся в их «усилении». Каждая обнаружившая коллизию станция передает специальный jam-сигнал некоторой длительности (значение стандартом не регламентируется). Jam-сигнал выполняет две важные функции. Во-первых, является признаком возникновения коллизии, что позволяет другим станциям сразу «увидеть» коллизию (столкнувшиеся передатчики, выставившие jam-сигнал, и так знают о коллизии). Во-вторых, позволяет синхронизировать время начала отсчетов случайных задержек.

****

**Код лабораторной работы**

**Файл CollisionController.java:**

for (byte symbol: line) {  
 StringBuilder collisions = new StringBuilder();  
 int attempts = 0; //обнулить счетчик попыток  
 boolean sending = true;  
 if (!isPackageMode()) { //пакетный режим?  
 while (sending) { //пока кол-во попыток <10  
 if (isChannelFree()) { //канал свободен?  
 send(symbol); //передать кадр  
 sleep(*COLLISION\_DURATION*); //выждать окно коллизий  
  
 if (isCollision()) { //коллизия обнаружена?  
 collisions.append(*COLLISION\_SYMBOL*); //отправить джем-сигнал  
 attempts += 1; //инкрементировать счетчик попыток  
  
 if (attempts >= *MAX\_ATTEMPTS*) { //попыток слишком много?  
 sending = false;  
 } else {  
 Random rand = new Random();  
 int k = Math.*min*(attempts, *MAX\_ATTEMPTS*);  
 int r = rand.nextInt((int) Math.*pow*(2, k));  
 sleep(r \* *SLOT\_TIME*); //вычислить и выждать случайную задержку  
 }  
 } else {  
 *runOnUIThread*(() -> {  
 outputZone.appendText((char) symbol + "");  
 debugZone.appendText("Line" + i++ + " ");  
 //i++;  
 debugZone.appendText(collisions + "\n");  
 });  
 sending = false;  
 }  
 }  
 }  
 }  
}