Учреждение образования

Белорусский государственный университет

информатики и радиоэлектроники

Кафедра электронных вычислительных машин

Лабораторная работа №4

**СЛУЧАЙНЫЕ МЕТОДЫ ДОСТУПА К МОНОКАНАЛУ**

Выполнил: Проверил:

ст. гр. №950503 Марцинкевич В.А.

Гуринович А. В.

Минск 2021

**1**. **Коллизии.**

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.

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.

**2**. **Способы борьбы с коллизиями.**

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.

**3. CSMA/CD (Carrier Sense Multiple Access with Collision Detection).**

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.

****

Рисунок 1 – Алгоритм CSMA/CD

**4. Pseudoterminal**

class Pseudoterminal {

private:

int descriptor;

std::string port;

PseudoterminalSettings \*settings;

int counter;

std::map<unsigned int, std::pair<std::string, int>> lnetwork;

std::stringstream buffer;

public:

Pseudoterminal();

~Pseudoterminal();

bool create\_port();

std::string get\_port\_name();

bool connect(const std::string &port);

bool accept\_connect(const std::string &port);

void disconnect(const unsigned int &device);

void accept\_disconnect(const unsigned int &device);

std::map<unsigned int, std::pair<std::string, int>> get\_list\_network() const;

size\_t get\_count\_connect() const;

std::string read\_port(const std::size\_t &size);

size\_t write\_port(const std::string &str, const unsigned int &device);

void resending(const std::string &port);

void collision();

void change\_speed\_in(const size\_t &speed);

void change\_speed\_out(const size\_t &speed);

void close\_port();

bool is\_open();

int find\_device(const std::string &port) const;

private:

void init\_port\_settings();

void flush\_port\_buffer();

void wait() const noexcept;

// void error(std::string msg);

};

**5. Frame**

const size\_t MAX\_SIZE\_FRAME\_DATA = 150;

const size\_t FRAME\_ADRESS\_SIZE = 13;

struct Frame {

long int preamble;

char SFD;

char DA[FRAME\_ADRESS\_SIZE];

char SA[FRAME\_ADRESS\_SIZE];

size\_t size;

char data[MAX\_SIZE\_FRAME\_DATA];

size\_t FCS;

template<class Archive>

void save(Archive &ar, const unsigned int version) const {

ar & this->preamble;

ar & this->SFD;

ar & this->DA;

ar & this->SA;

ar & this->size;

ar & this->data;

ar & this->FCS;

}

template<class Archive>

void load(Archive &ar, const unsigned int version) {

ar & this->preamble;

ar & this->SFD;

ar & this->DA;

ar & this->SA;

ar & this->size;

ar & this->data;

ar & this->FCS;

}

BOOST\_SERIALIZATION\_SPLIT\_MEMBER()

friend class boost::serialization::access;

friend std::ostream& operator<<(std::ostream &out, struct Frame &F) {

out << "\t---FRAME---\n";

out << "Preamble: " << F.preamble << "\n";

out << "SFD: " << F.SFD << "\n";

out << "DS: " << F.DA << "\n";

out << "SA: " << F.SA << "\n";

out << "Size: " << F.size << "\n";

out << "Data: " << F.data << "\n";

out << "FCS: " << F.FCS << "\n";

out << "\t---END FRAME---\n";

return out;

}

};

**6. Cropping**

const size\_t MAX\_SIZE\_FRAME = sizeof(struct Frame) \* 3;

class Cropping {

private:

struct Frame value;

public:

constexpr Cropping() : value{} {}

void start() noexcept;

char\* get\_data() noexcept;

template <typename Type>

void change\_data(Type &data) {

}

template<class Archive>

void save(Archive &ar, const unsigned int version) const {

ar & this->value;

}

template<class Archive>

void load(Archive &ar, const unsigned int version) {

ar & this->value;

}

BOOST\_SERIALIZATION\_SPLIT\_MEMBER()

friend class boost::serialization::access;

friend std::ostream& operator<<(std::ostream &out, Cropping &C);

// friend std::istream& operator>>(std::istream &in, Cropping &C);

void set\_sender(const std::string &sender) noexcept;

void set\_recipiend(const std::string &recipiend) noexcept;

private:

void set\_preamble() noexcept;

void set\_SDF() noexcept;

void set\_size\_frame() noexcept;

const unsigned int fstart = 00101;

}