Многопоточность работа с сетью делегаты

Простые потоки

Пока что в локальной сети

ПОТОКИ ВЫПОЛНЕНИЯ

Поток выполнения это

Поток выполнения являет собой последовательность действий, выполняемый процессором. В наших предыдущих программах мы имели только один, главный поток, Который и представлял нашу программу.

Но мы можем сами, когда захотим, определить новый поток, с его действиями, и запустить его. Эти действия начнут выполняться параллельно с основными действиями.

На кой нужно несколько потоков?

Обычное приложение имеет один поток выполнения, и выполняет операции друг за другом. Если необходимо сделать несколько действий одновременно, одно всегда будет выполнено раньше другого, из-за последовательности операций.

Если несколько операций должны работать параллельно и независимо друг от друга, мы можем вынести их в разные потоки, Так что действия из первой операции будут выполняться одновременно с действиями второй операции.

Это очень полезно когда операция значимую часть свою времени чего-то ожидает, ответ от сервера, из БД, диска, откуда угодно.

Например, для работы с сетью мы должны в любой момент времени принять сообщение, и так же в любой момент времени должны иметь возможность отправить сообщение в сеть

Гонка ресурсов, что это и зачем?

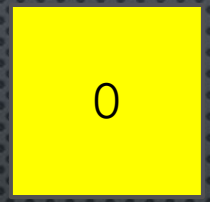
Допустим у нас стоит задача увеличить некоторую переменную на 100.

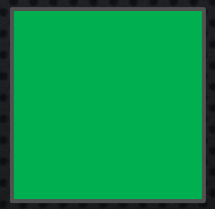
И мы не напишем += 100, мы будем увеличивать ее на единицу 100 раз.

А для того чтобы этот процесс произошел как можно быстрее, мы будем делать это в двух потоках.

А еще, мы будем не просто увеличивать переменную, а копировать ее значение, увеличивать на 1, а потом увеличенное значение записывать в переменную. Что может пойти не так?

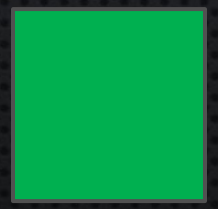
РАССМОТРИМ, КАК ЭТО РАБОТАЕТ



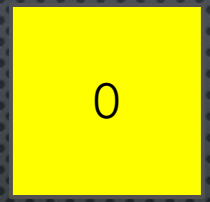


Поток 1

Поток 2



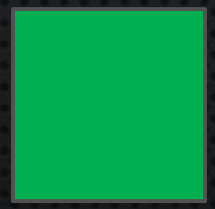
РАССМОТРИМ, КАК ЭТО РАБОТАЕТ



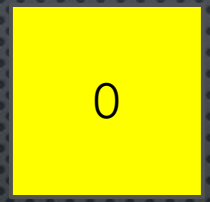
Считали значение 0

Поток 1

Поток 2



РАССМОТРИМ, КАК ЭТО РАБОТАЕТ

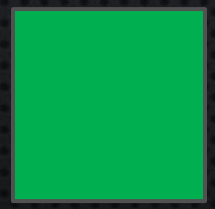


Увеличили его на 1

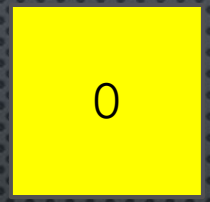
Считали значение 0

Поток 1

Поток 2



РАССМОТРИМ, КАК ЭТО РАБОТАЕТ



Увеличили его на 1

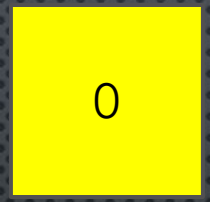
Считали значение 0

Поток 1

Считали значение 0

Поток 2

РАССМОТРИМ, КАК ЭТО РАБОТАЕТ



Увеличили его на 1

Считали значение 0

Поток 1

Увеличили его на 1

Считали значение 0

Поток 2

РАССМОТРИМ, КАК ЭТО РАБОТАЕТ

Записали 1 в результат

Увеличили его на 1

Считали значение 0

Поток 1

Увеличили его на 1

Считали значение 0

Поток 2

РАССМОТРИМ, КАК ЭТО РАБОТАЕТ

Записали 1 в результат

Увеличили его на 1

Считали значение 0

Поток 1

Записали 1 в результат

Увеличили его на 1

Считали значение 0

Поток 2

КАК ИТОГ – ПРОШЛО ДВЕ ИТЕРАЦИИ НАШЕГО АЛГОРИТМА, А ВОТ ЗНАЧЕНИЕ УВЕЛИЧИЛОСЬ ЛИШЬ НА 1

1

Записали 1 в результат

Увеличили его на 1

Считали значение 0

Поток 1

1

Записали 1 в результат

Увеличили его на 1

Считали значение 0

Поток 2

1

Или более реальная задача, мы хотим выводить два типа сообщений разным цветом.

Один поток будет выводить Hi red world красным цветом

А второй поток будет выводить Hi yellow world желтым цветом

Соответственно каждый поток выполняет по 4 действия

1. Запомнить текущий цвет текста, чтобы вернуть все обратно
2. Поменять цвет текста на свой
3. Вывести текст
4. Поменять цвет текста на исходный

Наша консоль. Изначально цвет текста белый, допустим

Поток 1, пишет красным

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Запомнили цвет текста

Поток 1, пишет красным

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Запомнили цвет текста

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Поменяли цвет текста на желтый

Запомнили цвет текста

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Hi red world

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Поменяли цвет текста на желтый

Запомнили цвет текста

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Hi red world

Поменяли цвет на исходный

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Поменяли цвет текста на желтый

Запомнили цвет текста

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Hi red world

Hi yellow world

Поменяли цвет на исходный

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Вывели текст

Поменяли цвет текста на желтый

Запомнили цвет текста

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Hi red world

Hi yellow world

Поменяли цвет на исходный

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Поменяли цвет на исходный

Вывели текст

Поменяли цвет текста на желтый

Запомнили цвет текста

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый, допустим

Hi red world

Hi yellow world

В ИТОГЕ И НЕ ТЕМИ ЦВЕТАМИ ВЫВЕЛИ, И ЕЩЕ И ИЗНАЧАЛЬНЫЙ ЦВЕТ НА МЕСТЕ НЕ ОСТАЛСЯ ИСХОДНЫЙ ☹

Поменяли цвет на исходный

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Поток 1, пишет красным

Поменяли цвет на исходный

Вывели текст

Поменяли цвет текста на желтый

Запомнили цвет текста

Поток 2, пишет желтым

ПРОВЕРИМ

Обозначим проблему

В обоих алгоритмах мы надеялись, что итерация каждого из них пройдет последовательно, и доступа к общему ресурсу во время выполнения не будет. И правда, ведь если сначала один поток сделает свой набор действий, а уже потом начнет работать другой – все будет в порядке.

Нам необходимо, чтобы на вот этих участках кода единовременно мог находиться только один поток.

Решение - lock

Lock позволяет какому-то потоку ”захватить” так называемый примитив синхронизации, чтобы ни один другой поток не мог получить доступ к этому примитиву, и соответственно не мог зайти в участок когда, в котором может находиться только один поток.

Наша консоль. Изначально цвет текста белый

Примитив синхронизации

Поток 1, пишет красным

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый

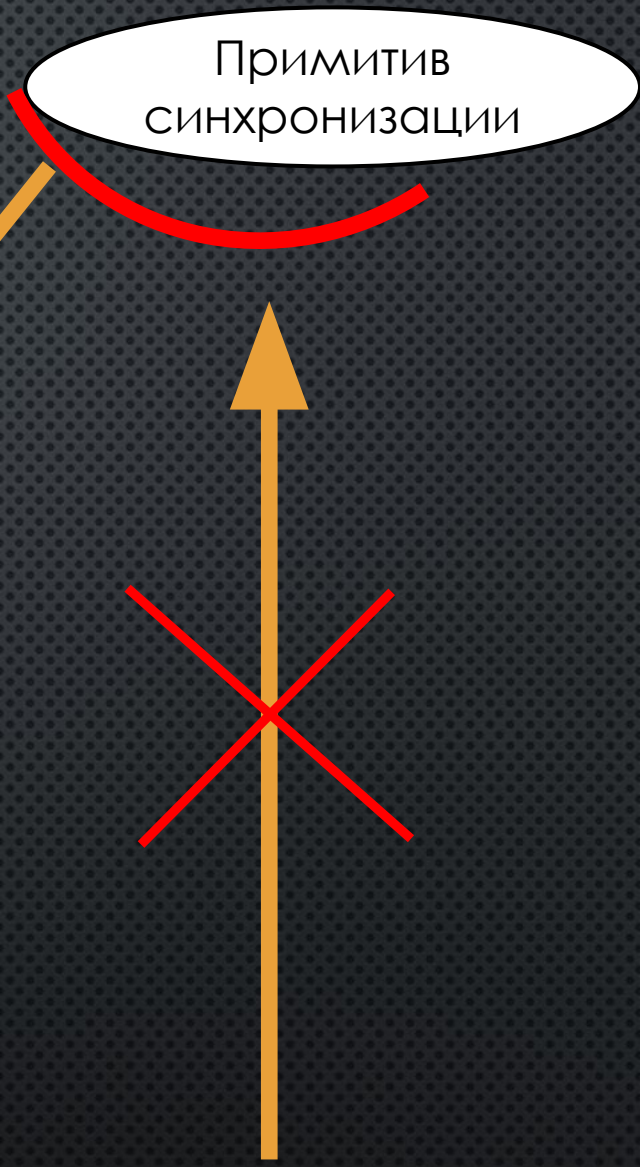
Примитив синхронизации

Захватили примитив синхронизации

Поток 1, пишет красным

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый



Захватили примитив синхронизации

Поток 1, пишет красным

Попытались захватить примитив

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый

Примитив синхронизации

Запомнили цвет текста

Захватили примитив синхронизации

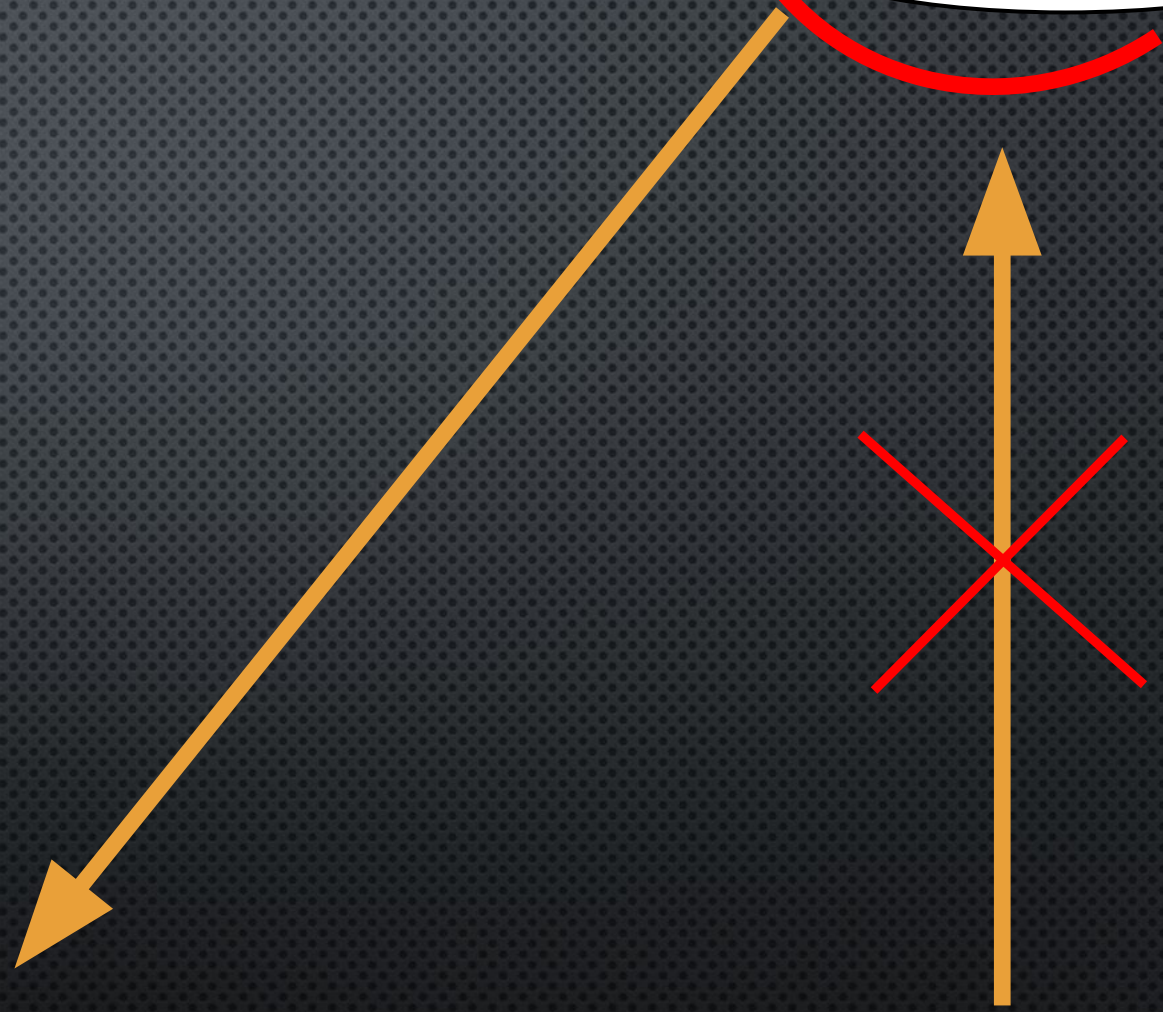
Поток 1, пишет красным

Попытались захватить примитив

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый

Примитив синхронизации



Сменили цвет консоли на красный

Запомнили цвет текста

Захватили примитив синхронизации

Поток 1, пишет красным

Попытались захватить примитив

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый

Hi red world

Примитив синхронизации

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Захватили примитив синхронизации

Поток 1, пишет красным

Попытались захватить примитив

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый

Hi red world

Примитив синхронизации

Поменяли цвет на исходный

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Захватили примитив синхронизации

Поток 1, пишет красным

Попытались захватить примитив

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый

Hi red world

Примитив синхронизации

Отпустили примитив синхронизации

Поменяли цвет на исходный

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Захватили примитив синхронизации

Поток 1, пишет красным

Попытались захватить примитив

Поток 2, пишет желтым

Наша консоль. Изначально цвет текста белый

Hi red world

Hi yellow world

Примитив синхронизации

Отпустили примитив синхронизации

Поменяли цвет на исходный

Вывели текст

Сменили цвет консоли на красный

Запомнили цвет текста

Захватили примитив синхронизации

Поток 1, пишет красным

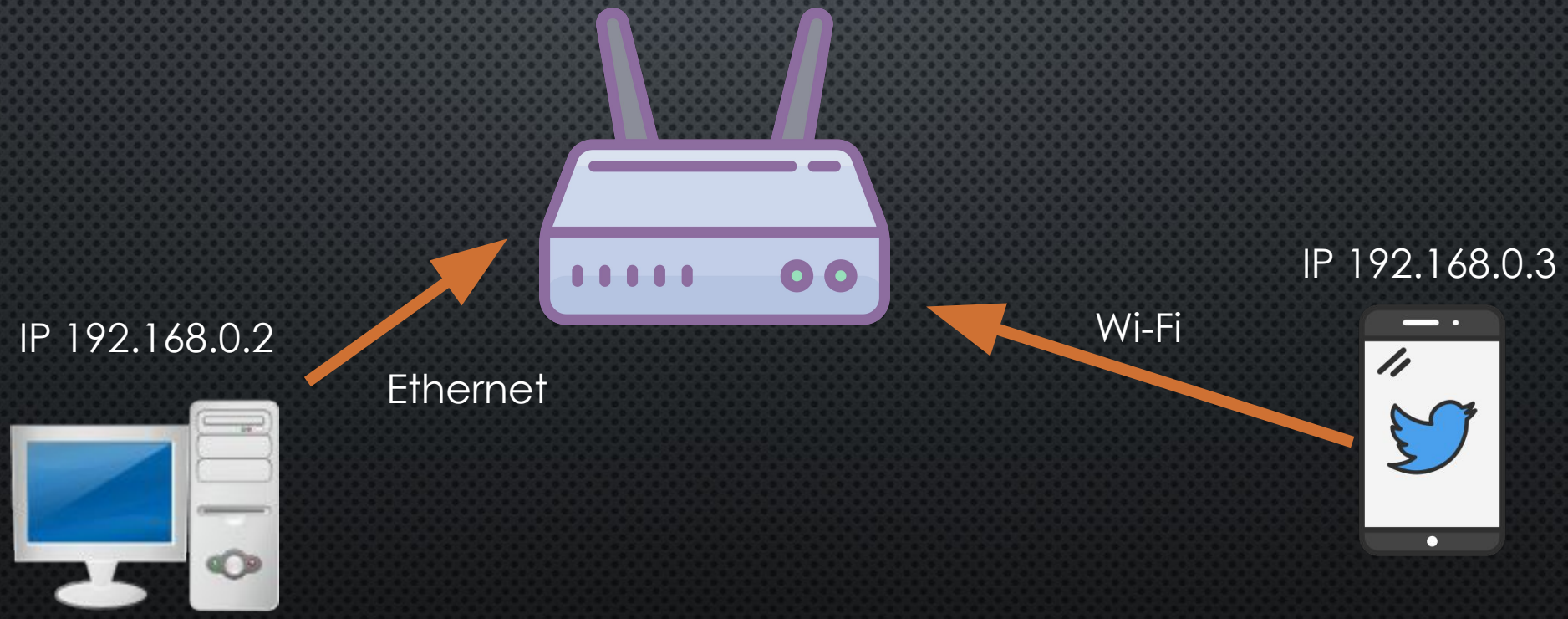
Ну и выполнили все действия

Дорвались до примитива

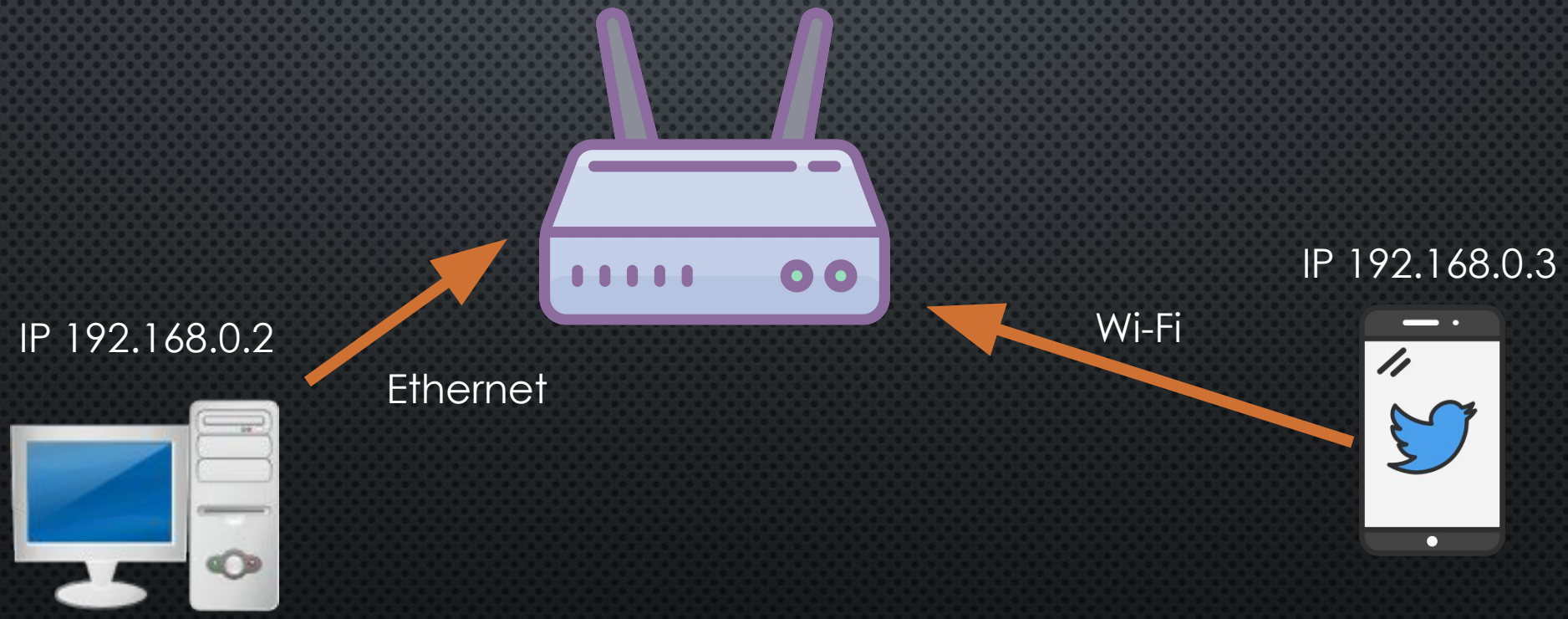
Поток 2, пишет желтым

LOCK ПОЗВОЛЯЕТ ОБЕЗОПАСИТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ПОСЛЕДСТВИЙ

КАЖДОЕ УСТРОЙСТВО, ПОДКЛЮЧЕННОЕ К ХОТЬ КАКОЙ СЕТИ ИМЕЕТ IP АДРЕС



КАЖДОЕ УСТРОЙСТВО, ПОДКЛЮЧЕННОЕ К ХОТЬ КАКОЙ СЕТИ ИМЕЕТ IP АДРЕС



Мы говорим именно локальной сети, а не глобальном интернете, сегодня рассмотрим общение между устройствами именно в локальной сети.

СЕГОДНЯ СМОТРИМ НА ТРАНСПОРТНЫЙ УРОВЕНЬ

А именно на протокол UDP

	Единица данных	Уровень	Функция	Примеры протоколов
ОС	Поток	Прикладной	Прикладная задача	HTTP, SMTP, DNS, etc.
		Представления	Представление данных, шифрование, etc.	MIME, SSL
		Сеансовый	Взаимодействие хостов (на уровне ОС)	NetBIOS, именов. пайпы
	Сегмент	Транспортный	Соединение конец-в-конец, контроль передачи данных	TCP, UDP
Сеть	Пакет	Сетевой	Логическая адресация и маршрутизация пакетов	IP, ICMP
	Фрейм	Канальный	Физическая адресация	IEEE 802.3, ARP, DHCP
	Бит	Физический	Кодирование и передача данных по физическому каналу	IEEE 802.3

КАК ПРОИСХОДИТ ПЕРЕДАЧА ИНФОРМАЦИИ С UDP

IP 192.168.0.2

- Google chrome
- WoW
- Наша программа
- OS
- Антивирус
- Яндекс музыка

Порт

- 5456
- 7856
- 7851
- 8954
- 45
- 905

???

- 4536
- 8394
- 94
- 327
- 2

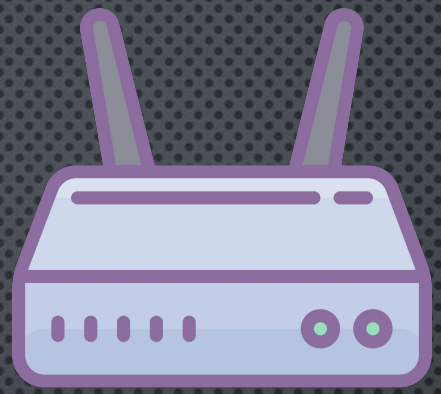
- Наша прога
- PUBG
- OneNote
- Slack
- Outlook

IP 192.168.0.3

Дейтаграмма

- IP назначения
- Порт назначения
- IP отправителя
- Порт отправителя
- Пользовательские данные

Программа формирует дейтаграмму
К 192.168.0.3
К 5001
От 192.168.0.2
От 5000
Данные – строка «PING»



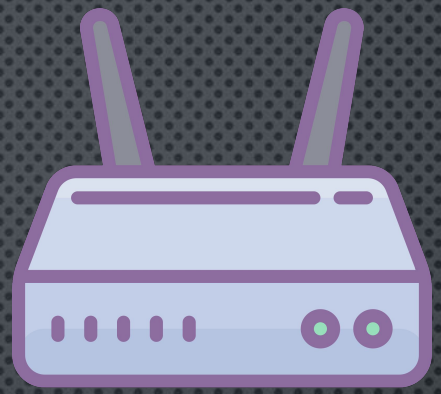
IP 192.168.0.3

Наша программа, порт 5001

IP 192.168.0.2

Наша программа, порт 5000

Эта хреновина отправляется на местный роутер



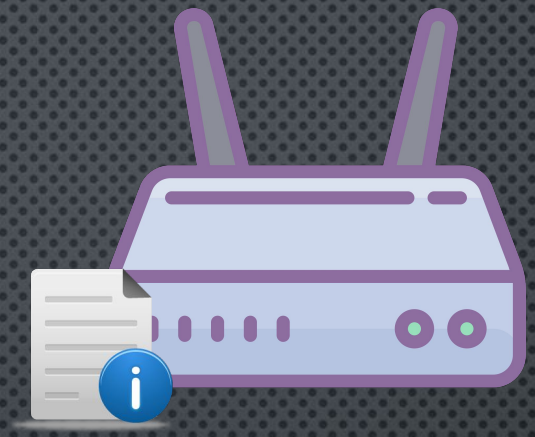
IP 192.168.0.3

Наша программа, порт 5001

IP 192.168.0.2

Наша программа, порт 5000

Роутер знает, где находится ПК с необходимым IP



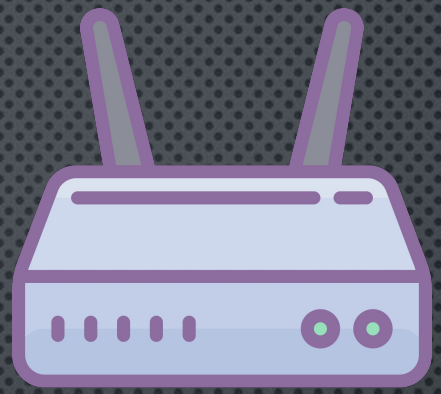
IP 192.168.0.3

Наша программа, порт 5001

IP 192.168.0.2

Наша программа, порт 5000

А компьютер знает, у какой программы нужный порт, и отправляет наши данные именно ей



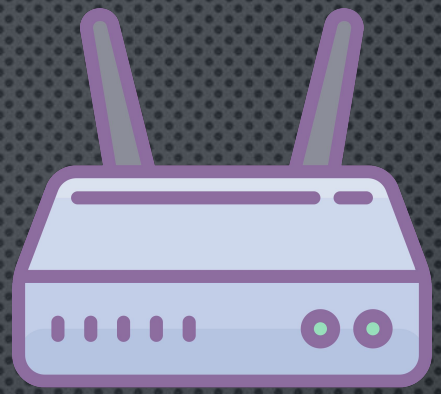
IP 192.168.0.3

Наша программа, порт 5001

IP 192.168.0.2

Наша программа, порт 5000

Программ а на втором компьютере, получив информацию PING и зная, кто ее отправил, формирует обратную посылку с данными «PONG», и отправляет ее.



IP 192.168.0.3

IP 192.168.0.2

Наша программа, порт 5000

Наша программа, порт 5001

Для нашего базового Ping Pong

Основной поток должен считывать слова из консоли, и отправлять их по сети

Второй поток, который мы запустим, будет принимать сообщения, и выводить их на консоль.