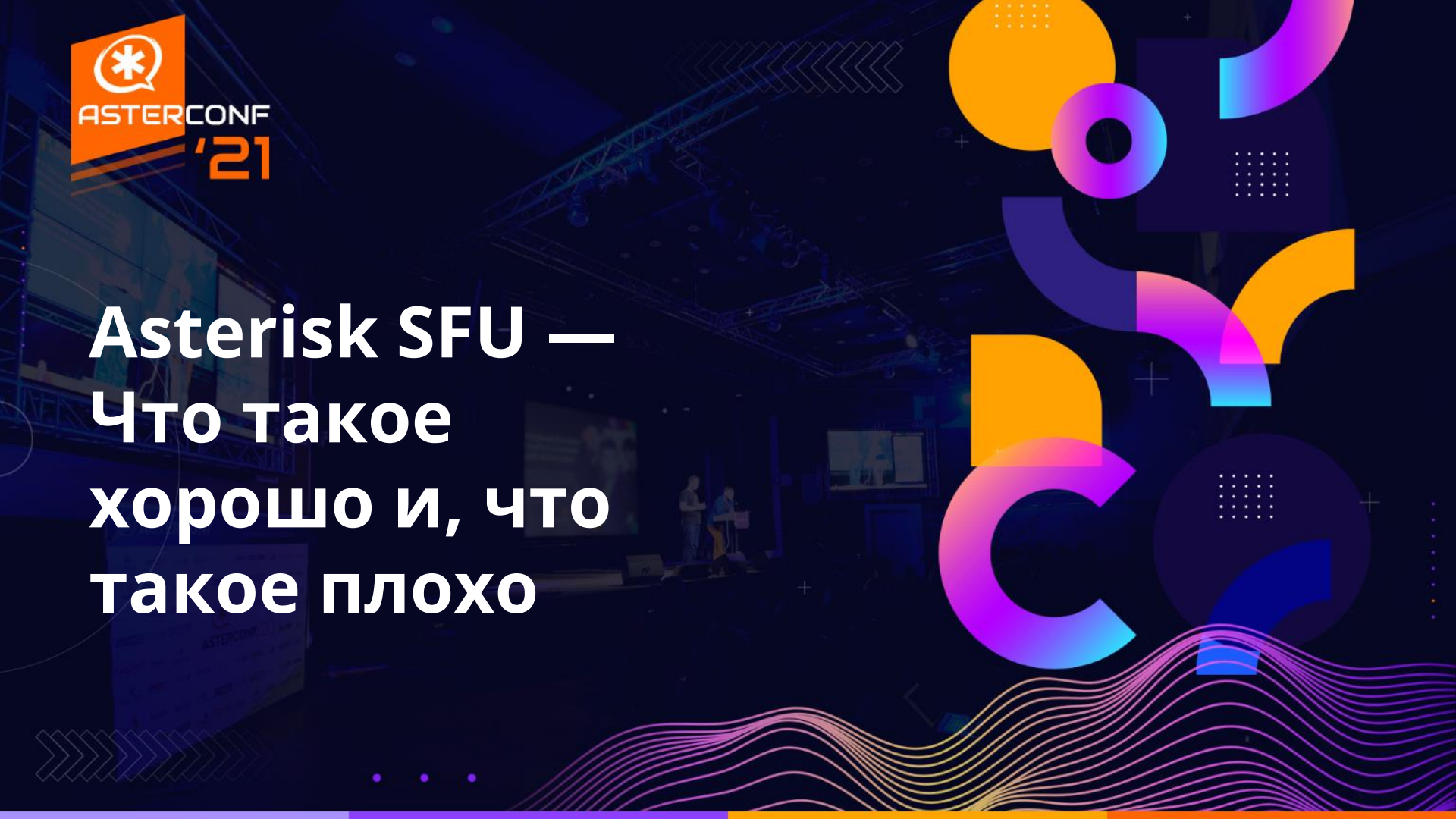
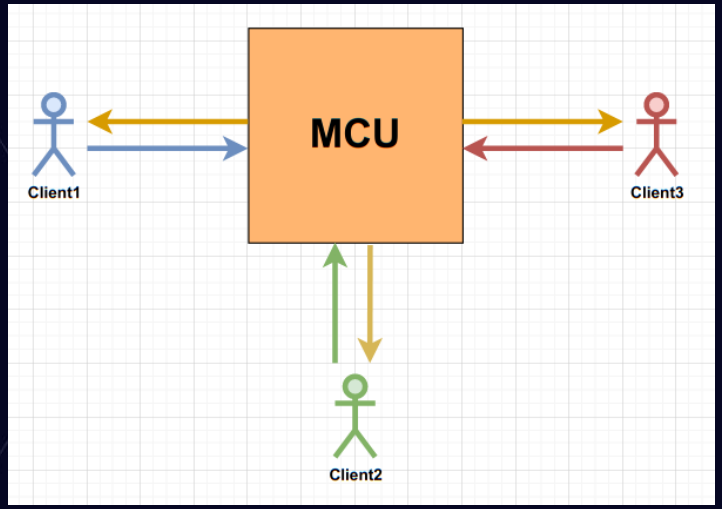


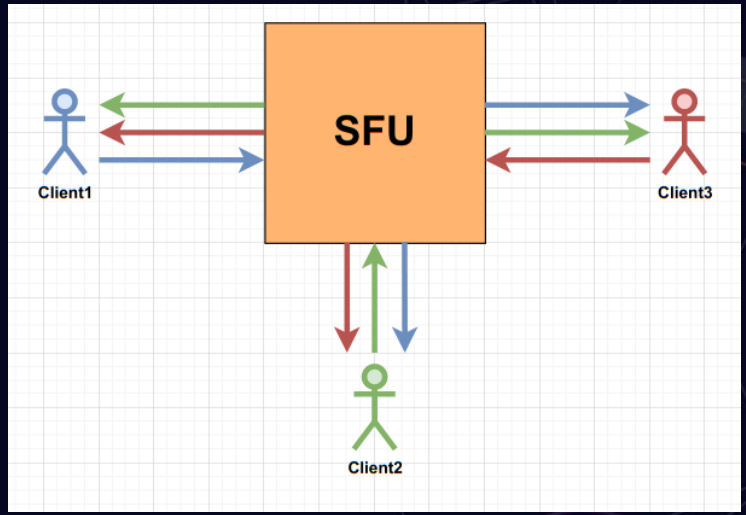
Asterisk SFU — Что такое хорошо и, что такое плохо

Что такое SFU?

- **S**ELECTIVE **F**ORWARDING **U**NIT
- Media Router – маршрутизирует входящий медиа поток на подписавшихся на него пользователей
- Не микширует медиа потоки
- Не транскодирует медиа из одного кодека в другой

MCU vs SFU





Плюсы и минусы MCU

- Плюсы
- Относительно маленькая сетевая нагрузка
- Готовое к администрированию ПО
- Поддерживает множество железных клиентов

- Минусы
- Очень серьезная нагрузка CPU/GPU
- Фактическая невозможность кластерного расширения
- Невозможность манипулирования конечной картинкой
- Сложность администрирования
- Сложность кода

Плюсы и минусы SFU

- Плюсы
- Слабая нагрузка на сервер
- Легкость масштабирования и кластеризации
- Отображение сетки — полностью клиентская забота
- Простота администрирования

- Минусы
- Повышенные требования к полосе пропускания
- Больше библиотека, чем готовое ПО
- Необходимость понимания процессов происходящих в realtime потоке данных
- Мало готовых железных клиентов

Почему в Asterisk нет MCU, но есть SFU

- MCU объективно сложнее реализовывать
- SFU базово проще
- Микширование отдано на сторону клиента
- Asterisk стремится быть WebRTC complaint продуктом

Нюансы SFU

- Так как **mod_sfu** прежде всего был имплементирован для видео, он должен решать следующие вопросы :
- Уметь оптимизировать использование канала данных
- Управлять качеством медиа для получателя

Оптимизация канала данных

- Чтобы оптимально использовать канал данных необходимо знать предполагаемую пропускную полосу (estimated bitrate) как отправителя так и получателя.

- Методы:

- RTCP Receiver Report сообщения
- RTCP REMB
- Sender Side Bitrate Estimation Mechanisms (Например, Transport-CC: RFC 8888)

RTCP Receiver Report

- можно получить кол-во потеряных пакетов и посчитать изменение пропускной способности принимающей стороны:
- Пример (очень упрощенно):
  Шлем **640**(px) *** 480**(px) *** 15**(frames) *** 0,08**(moving coef) **~ 369 Kb/s**
- RR отвечает: я недосчитался 20 фреймов за 4 секунды.
- (15 * 4 — 20)/4 = **10** frames/s;
  10 * 640 * 480 * 0,08 ~ **246 Kb/s**
- **246 < 369** → полоса просела, уменьшаем качество картинки

- **Минус:** репорты содержат фактическую информацию об утере пакетов
- **Следствие:**
  - понимание, что полоса просела происходит сильно постфактум.
- - пользователь какое то время видит рваную/статичную картинку (freezes)

RTCP Receiver **E**stimated **M**ax **B**itrate **(REMB)**

- Нотация в **SDP**: a=rtcp-fb:<payload type> **goog-remb**
- **Задача:** Слать **Estimated Max Bitrate** получателя в определенный интервал (на практике: ± 1 сек)
- Получатель медиа анализирует задержки между пакетами и на основе этих данных генирирует **REMB** сообщение
- Получатель **REMB** определяет имеет ли смысл менять качество медиа

- **Минусы:**
- Никогда не был принят как стандарт (Expired : 2014)
- Deprecated в Chrome (правда в этом статусе он уже 4 года)

RTCP Transport-wide congnestion control

- Нотация в **SDP**: a=rtcp-fb:<payload type> **transport-cc**
- Задача: Слать RTCP feedback (да да, еще один) со списком ожидаемых пакетов и их статус (пришел, не пришел, сильно ли опоздал)
- Отправитель решает — переотправить пакет или подождать
- Легче поддерживать код т. к. вся вычислительная логика только на стороне отправителя
- Позволяет реализовать такие алгоритмы как: I-D.ietf-rmcat-gcc, I-D.ietf-rmcat-nada,I-D.ietf-rmcat-scream-cc

- **Минусы:**
- Есть RFC 8888, но он не реализован. На практике используется драфт **draft-holmer-rmcat-transport-wide-cc-extensions-01**, который Expired: 2017

Поддержка оптимизации канала данных в Asterisk

- RTCP Receiver Report сообщения ←---- ... *Атсутсвуит!
- RTCP REMB ←----- **Здесь, тааришщ сержант!**

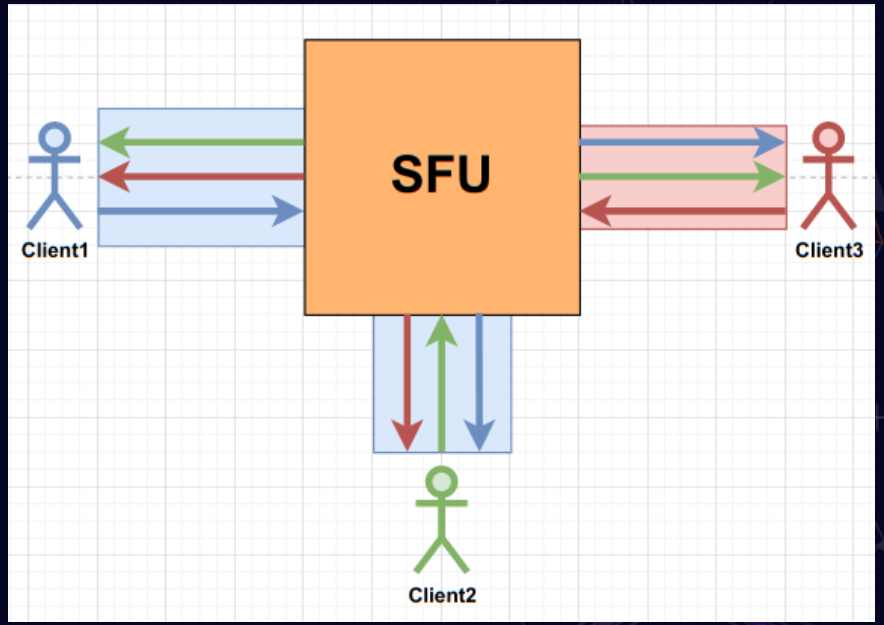
```
1288        /* We only care about REMB reports right now. In the future we may be able to use sender or
1289         * receiver reports to further tweak things, but not yet.
1290         */
1291        if (frame->subclass.integer != AST_RTP_RTCP_PSFB || feedback->fmt != AST_RTP_RTCP_FMT_REMB ||
1292                bridge->softmix.video_mode.mode != AST_BRIDGE_VIDEO_MODE_SFU ||
1293                !bridge->softmix.video_mode.mode_data.sfu_data.remb_send_interval) {
1294                return;
```

- Sender Side Bitrate Estimation Mechanisms ←----- ... *Атсутсвуит!

* Вообще присутсвует, но не имеет отношения к анализу пропускной способности

Проблема Estimated Bitrate

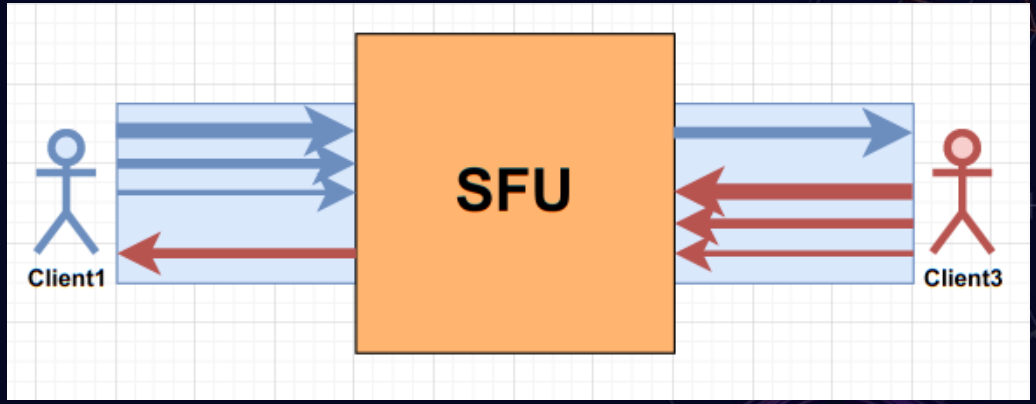
- 3 клиента общаются, у одного из них проседает пропускная полоса
- Как на это реагировать отправителям?
- Как на это реагировать получателям?
- Как на это реагировать SFU?



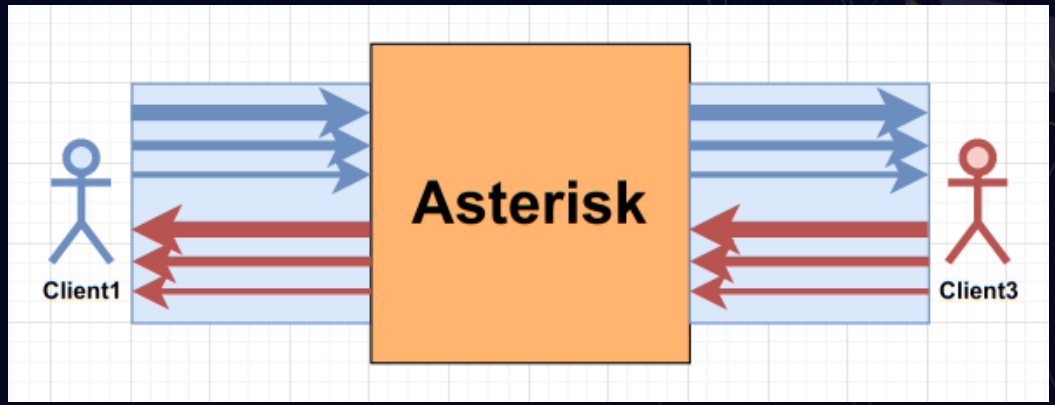
Simulcast

- Отправитель шлет несколько потоков данных разной кондиции, а получатель в праве выбирать — какую кондицию он хочет получить

- Реализовано в VP8 как несколько видео трэков
- Реализовано в H264 и VP9 как SVC (scalable video coding)

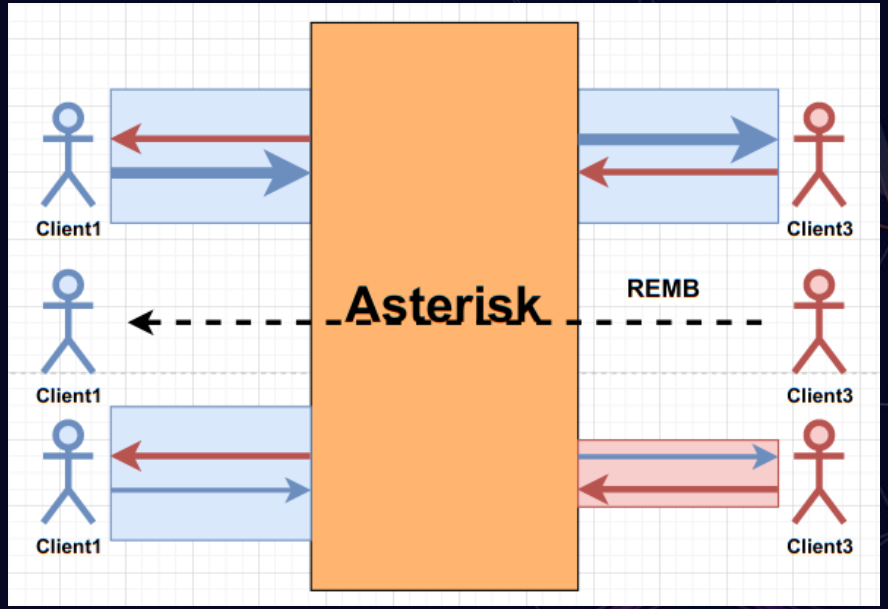


Как это реализовано в Asterisk



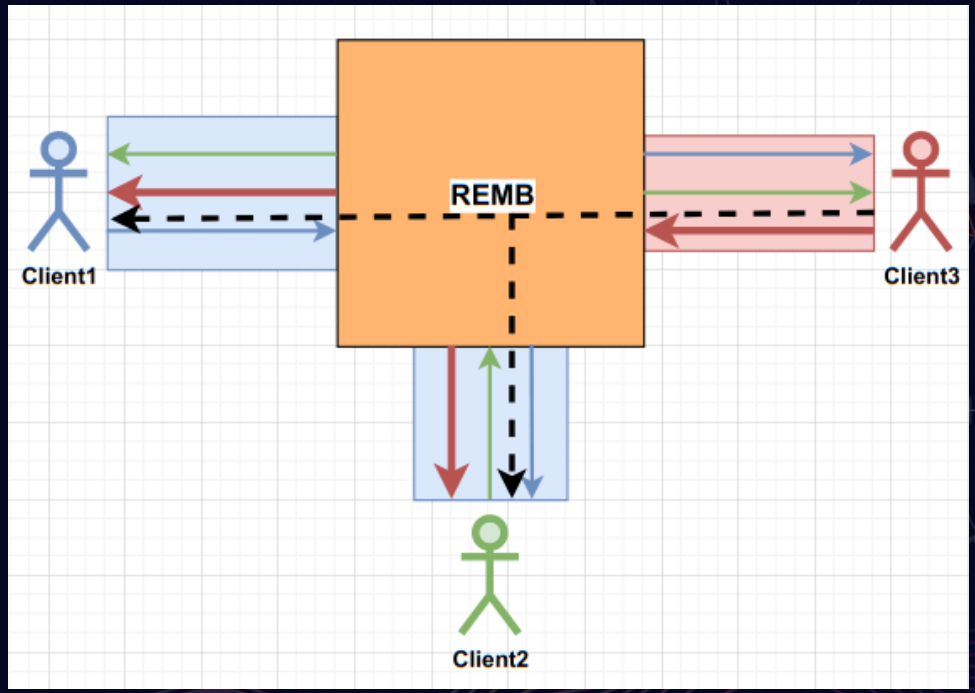
Как это реализовано в Asterisk

- Получатель шлет REMB
- Астериск пересылает пакет отправителю
- Отправитель изменяет
- качество видео
- Получатель теперь получает видео меньшего качества, но подходящее по полосе пропускания получателя



Как это реализовано в Asterisk

- Получатель шлет REMB
- Астериск пересылает REMB отправителям
- Отправители изменяют качество видео
- Получатель принимает видео меньшего качества
- Другие получатели ТОЖЕ получают видео меньшего качества



Что сделано хорошо Asterisk SFU

- SFU Mode — сделан для видео, соотвественно это никак не ограничивает нас пользоваться преимуществами аудио конференции:
  - Аннонсирование присоединившегося участника
  - Упарвление конференцией
  - Профилирование

Спасибо за внимание!

У кого-нибудь есть вопросы?

iurii@devrtc.tech

devrtc.tech