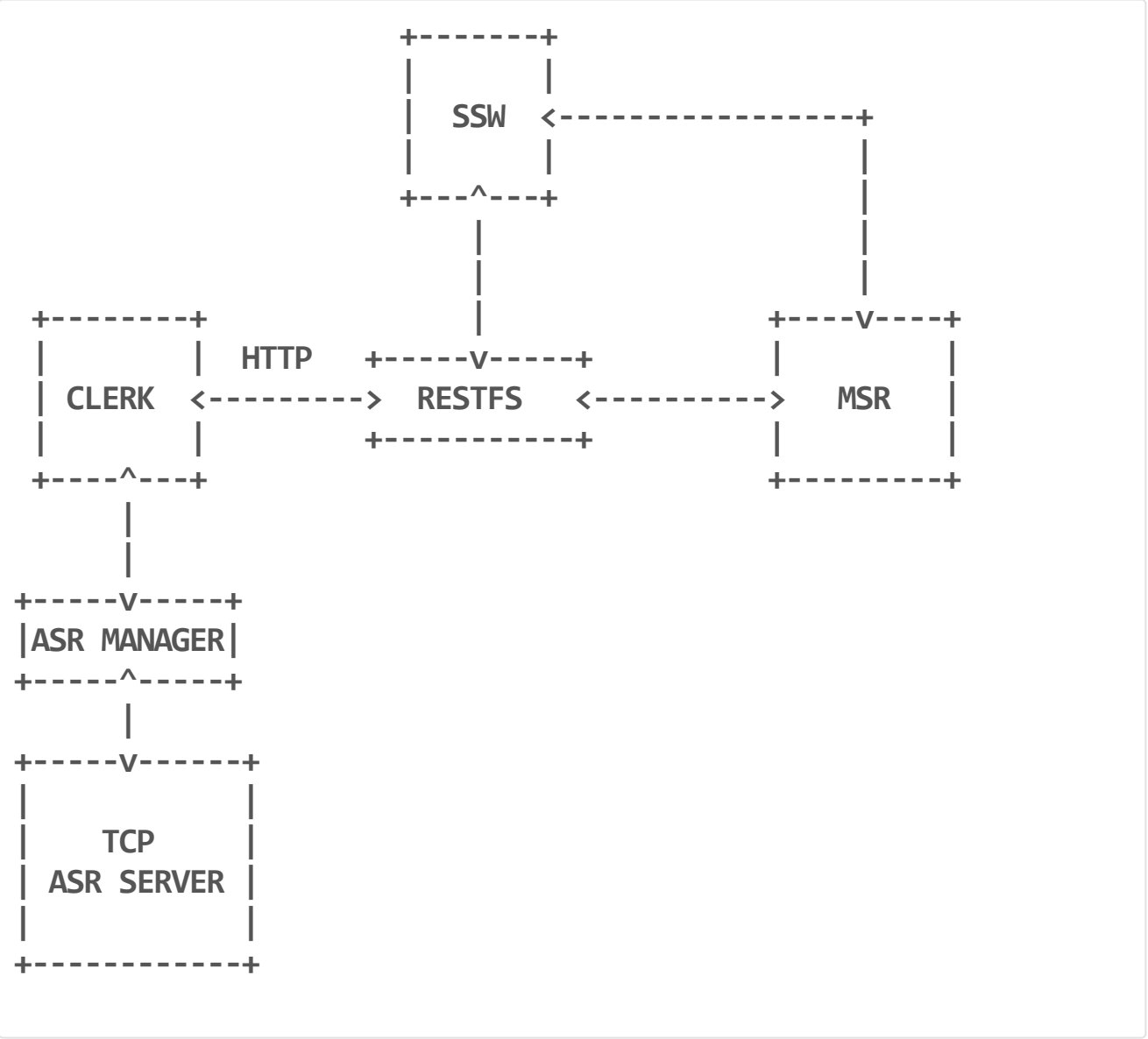
Wiki »

Архитектура и взаимодействие с SSW



Сразу разберу непонятные названия слева:

- Clerk - он же ecss-clerk или автосекретарь, планировался как прокси сервер между asr и ядром. HTTP сервер (elixir/phoenix) принимающий запросы на распознавание и отдающий результаты в виде JSON;
- Asr Manager - HTTP сервер (python), который управляет TCP инстансами Kaldi. Может пересобрать модель, добавив новые слова, держать несколько экземпляров tcp серверов и управлять ими;
- TCP Asr Server - собственно, один из инстансов, которым управляет Asr Manager

Ecss-clerk

HTTP сервер принимающий запросы на распознавание речи в виде готовых pcm/wav файлов. Состоит из следующих основных компонентов:

- Pcm Manager - управляет пришедшими за распознавание PCM файлами. В нем хранятся аудиоданные, которые при необходимости можно отправить на распознавание или записать на диск;
- Asr Manager - компонент для взаимодействия с HTTP сервером Kaldi. При старте узнает статус, сохраняет адрес TCP сервера, которому позже отправляет на распознавание аудио. Также через него

запускается процесс рекомпиляции;
- Phonebook - осуществляет поиска в телефонной книге распознанной строки. При старте импортирует книгу в формате xml через restfs, обновляется каждые 12 часов;
- Database - приложение для взаимодействия с базой данных (mysql).

Как это работает

Юзер снимает трубку и набирает номер, на который завязан ivr скрипт автосекретаря. После синтезированного приветствия, в котором пользователя просят произнести имя и фамилию, поток с его микрофона отправляется на специальный порт MSR, который работает по следующему принципу: детектит тишину и нарезает по ней на чанки. Каждый чанк отправляется отдельным HTTP POST запросом в сторону ecss-clerk (через restfs) с заголовком "Content-type: Transfer-Encoding". Идентификатором запроса является путь HTTP запроса вида "/speech-recognition/domain/eltex/2021-01-25_18-30-18-866402_asr_5501-4893.wav". По имени wav файла (хотя на самом деле это чистый pcm) ecss-clerk и идентифицирует чанк. Каждый чанк отправляется на распознавания, по результату которого phonebook пытается найти подходящий номер. Если не удалось, то clerk просит следующий чанк и так до тех пор, пока не получится найти номер или не сработает таймаут в ivr скрипте.

В случае успеха json с номером летит на MSR через restfs, MSR отдает json ядру. Промежуточные ответы тоже доходят до ядра. Они имеют HTTP код 206 и не содержат в прилагаемом в теле json-е поля с номером, но имеют поле с распознанной строкой. Имея эту строку по таймауту ivr может синтезировать ответ для пользователя, уведомив о том, что найти распознанную строку не удалось. Если до таймаута ecss-clerk не пришлёт ответ ядру, то синтезированное уведомление не будет содержать распознанной строки.

Синтез происходит через restfs. Resfts проксирует запрос на настроенный tts сервер (пока это только yandex-tts).

После получения PCM ecss-clerk выполняет следующие действия:

- Отправляет pcm на распознавание (через TCP);
- По результату распознавания пытается найти номер в телефонной книге;
- Посылает ответ.

Ответ, как уже сказано выше, имеет формат json. Пример:

```
{
  "done": true,
  "negative_url": "negative/2021-01-25_18-30-18-866402_asr_5501-4893",
  "number": "4815",
  "positive_url": "positive/2021-01-25_18-30-18-866402_asr_5501-4893",
  "recognized": "антон черненко"
}
```

Поле **done** информирует о том, что распознавание закончено и номер был найден (больше задел на будущее, когда на asr будет проключаться поток с микрофона).
negative_url и **positive_url** - коллбеки, которые дергает ivr скрипт после получаения json. Помечают на ecss-clerk запрос как успешный или не успешный.
recognized - ответ от asr
number - номер

Пример промежуточного ответа:

```
{
  "done": false,
  "negative_url": "negative/2021-01-25_18-30-18-866402_asr_5501-4893",
  "recognized": "с детского алексеев"
}
```