Красный - IIS упал (No Respond (флаг Rst=true)), Красный (Ошибки протокола (HTTP Status != 200)), Серый – нет обмена, Жёлтый – наличие tcp пакетов (флаг Rst=false) Зелёный (HTTP Status = 200)– успешный обмен, Красный - обрыв сети отсутствие пинг.

Интерфейс

Контроллер

Библиотека

Входной блок анализатора (Сепаратор)

Анализатор TCP

Пакеты

TCP

Анализатор HTTP

HTTP

Измеритель временного интервала

Только Rst = 0

Событие о наличии/отсутствии трафика, или неопределённое состояние нет пакетов, или обрыв

Сигнал об отсутствии пакетов по истечении заданного времени

Ping генератор

Запуск от сигнала об отсутствии пакетов по истечении заданного времени

Останов генератора (Любой Rst (0 или 1))

Нет пинг (Break)

Внешний сигнал сброса ошибок

Событие о работе Сервисов

Счётчик Rst = 1

Открыть/Закрыть анализатор HTTP

Проверяем только выходные пакеты (Src = адрес хоста на котором установлен сервис). Также проверяем порт сервиса

Анализ трафика разделяется на 2 метрики:

1. Наличие трафика (Следят только за Rst)
2. Работа сервиса (Анализ HTTP ответа на наличие в нём ОК, или ошибки)

Сепаратор делит входной трафик на TCP и HTTTP потоки, которые подаются на соответствующие блоки анализа

Анализатор TCP:

1. Хранит предыдущее состояние
2. Имеет счётчик Rst = 1. Нужен, так как состояние Keep Alive сбрасывается сигналом Rst=1
3. Имеет состояния (Rst=1 (красный(Disconnect)), Rst=0 (Зелёный(Connect)), Sleep (отсутствие пакетов) (Жёлтый), Break (нет пинга) (Красный))
4. При получении Rst = 0
   1. Сбрасывает счётчик Rst = 1
   2. Останавливает Ping генератор.
   3. Проверяет предыдущее состояние, если предыдущее состояние отлично от Connect, то выдаёт на выход событие Rst=0 (Connect)
   4. Открывает Анализатор HTTP
5. При получении Rst = 1
   1. Проверяет, если счётчик Rst=1, равен 0, то устанавливает его в 1 и завершает работу.
   2. Иначе, если предыдущее состояние отлично от Disconnect, то выдаёт на выход событие Disconnect
   3. Закрывает Анализатор HTTP
6. При получении сигнала об отсутствии пакетов
   1. Если состояние не равно Connect, то ничего не происходит
   2. Сбрасывает счётчик Rst = 1
   3. Если предыдущее состояние отлично от Sleep, то выдаёт на выход событие Sleep
7. При получении сигнала Break
   1. Сбрасывает счётчик Rst = 1
   2. Если предыдущее состояние отлично от Break, то выдаёт на выход событие Break
   3. Закрывает Анализатор HTTP
8. При получении сигнала восстановление Break
   1. Если предыдущее состояние отлично от Break, то ничего не происходит
   2. Иначе выдаёт на выход событие Sleep

Анализатор HTTP:

1. Проверяет заголовок HTTP ответа
2. Хранит предыдущее состояние
3. Хранит общее кол-во ошибок предыдущего цикла обработки
4. Хранит кол-во ошибок для каждого клиента сервиса(метода)
5. Хранит описание ошибки для каждого клиента сервиса(метода)
6. Количество ошибок может быть сброшено внешним сигналом (например, из браузера)
7. ~~Имеет состояния (возможно только внутренние)~~
   1. ~~ОК (Зелёный), Errors (Красный), «Закрыт блоком Анализа TCP»~~
   2. ~~Работа блока начинается с состояния ОК~~
8. Выдаёт на выход событие «О работе сервиса» с указанием кол-ва ошибок по каждому клиенту и Сервису(Методу) и кодом ошибки.
9. При получении HTTP ОК
   1. Ничего не происходит
   2. ~~Если предыдущее состояние «Закрыт блоком Анализа TCP», то выдаёт на выход событие «О работе сервиса»~~
   3. ~~Иначе если предыдущее состояние отлично от ОК, и общее кол-во ошибок изменилось, то выдаёт на выход событие «О работе сервиса»~~
10. При получении HTTP Error,
    1. Увеличивает счётчик ошибок клиента сервиса(метода)
    2. Запоминает Сервис(Метод) вызвавший ошибку
    3. Вызывает событие «О работе сервиса» с передачей соответствующих параметров
    4. Рассчитывает общий счётчик ошибок
    5. Устанавливает состояние блока в Error
11. При получении сигнала «Сброса ошибок»
    1. ~~Если не равно «Закрыт блоком Анализа TCP», то ничего не происходит.~~
    2. ~~Иначе~~
       1. Сбрасывает счётчики ошибок всех клиентов и коды ошибок
       2. Сбрасывает общий счётчик ошибок
       3. Вызывает событие «О работе сервиса».
       4. Устанавливает состояние блока в ОК