Aceleração do Transporte de Dados com o Emprego de Redes de Circuito Dinâmicos (GT-ATER)

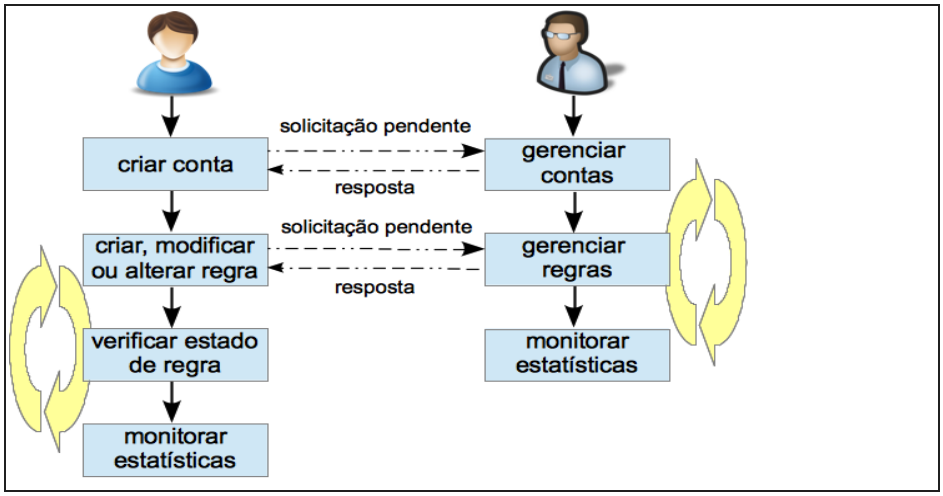
Fundamentos e Arquitetura

Neste documento, são descritos os fundamentos do serviço ATER, sua arquitetura básica e os tipos de regras que podem ser criadas e gerenciadas pelo serviço.

Visão Geral

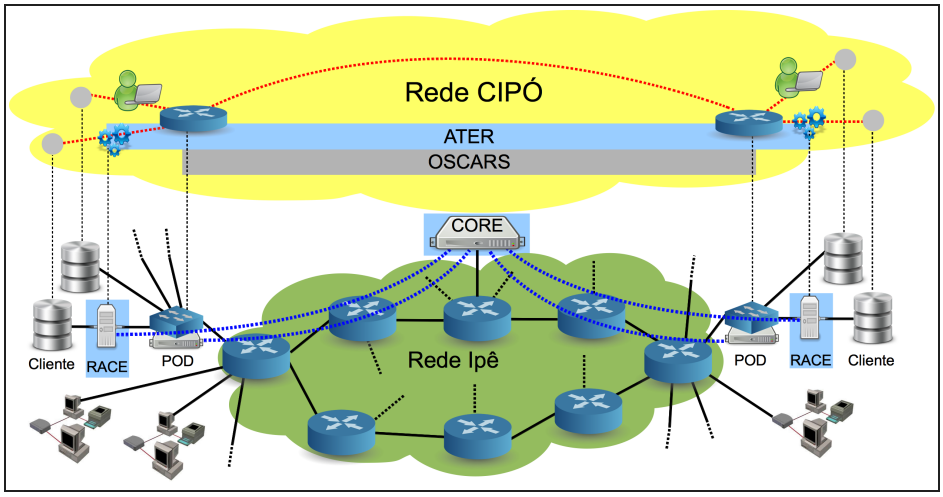
O GT-ATER tem como propósito a implementação de um serviço de transporte de grandes volumes de dados através de circuitos dinâmicos, os quais são criados automaticamente a partir de regras para identificação do perfil do tráfego. Nesse sentido, é objetivo do serviço enviar através de circuito dinâmico, criado sob demanda, o tráfego que combina com as regras definidas. O tráfego restante é enviado pela rede IP convencional.

As regras para identificação do perfil do tráfego podem ser definidas pelos clientes do sistema, inicialmente usuários de instituições que tenham necessidade de realizar transporte confiável de grandes fluxos de dados, como também por administradores pertencentes à equipe de operação da RNP. A Figura abaixo descreve as principais atividades de dois tipos de usuários (cliente e administrador) do ATER.



Para atingir seu objetivo, o ATER é construído sobre dois tipos de equipamentos. O primeiro, denominado **RACE** (*Rule Applier and Circuit Endpoint*), é um equipamento intermediário responsável por interceptar o tráfego do usuário, analisar seu perfil e, caso necessário, desviá-lo para que seja transferido pelo circuito dinâmico. Neste equipamento, ocorre a aplicação das regras para idenficação do perfil do tráfego e também a configuração e o estabelecimento da conectividade entre as extremidades dos circuitos dinâmicos. O segundo tipo de equipamento, denominado **CORE** (*Circuit Operation and Rule Establishment*), funciona como uma máquina servidora, hospedando o núcleo do serviço, ou seja, autenticação e autorização, gerenciamento de regras, armazenamento do resumo das estatísticas de tráfego, comunicação com o serviço de circuitos dinâmicos da RNP e o controle remoto dos RACEs.

Para criar o circuito dinâmico, o ATER faz uso do **Serviço Experimental CIPÓ** (SE-CIPÓ). Esse serviço permite criar circuitos dinâmicos no backbone RNP. Um dos principais componentes do SE-CIPÓ é o OSCARS, uma solução para aprovisionamento dinâmico de circuitos. O ponto de entrada para a rede CIPÓ é o POD, constituído minimamente por um servidor com OSCARS e um switch. A figura abaixo mostra como os equipamentos do ATER interagem com o POD do SE-CIPO.



Usuários

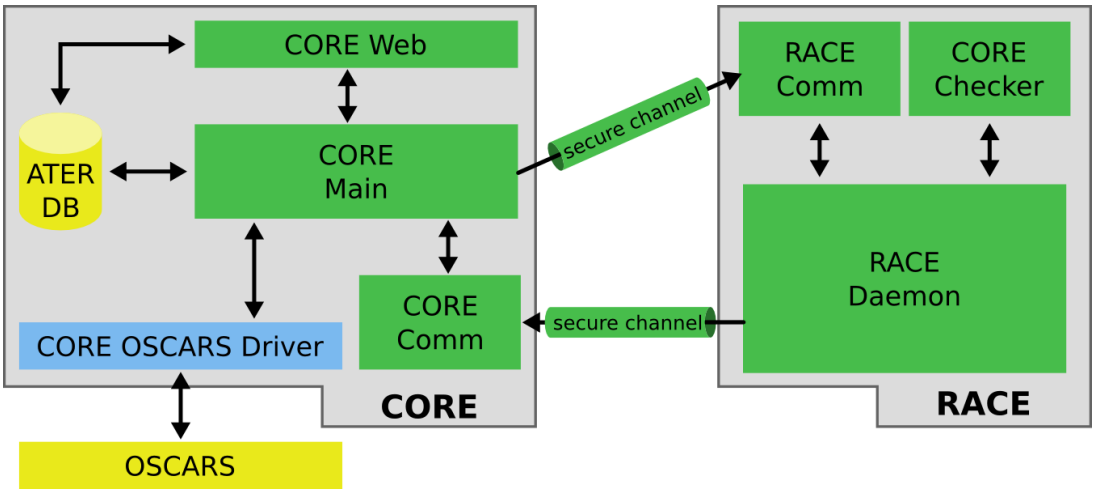
Os usuários do ATER podem ser classificados em três tipos: cliente, gerente local e administrador. A seguir são descritas as atividades que podem ser realizadas por cada usuário.

- **Cliente**:
  - Criação de conta
  - Criação de regras
  - Remoção de regras
  - Visualização de estatísticas de seus próprios circuitos
- **Gerente local**:

    - Todos os recursos do cliente
    - Aprovação de contas locais
    - Aprovação de regras locais
    - Criação/remoção de regras efetivas e de monitoramento
    - Visualização de estatísticas de regras efetivas e de monitoramento
- **Administrador**:
    - Todos os recursos do cliente
    - Aprovação de contas
    - Aprovação de regras
    - Criação/remoção de regras efetivas e de monitoramento
    - Visualização de estatísticas de regras efetivas e de monitoramento
    - Mudar o nível de um usuário para gerente local
    - Configuração do serviço

Arquitetura

A figura abaixo ilustra a arquitetura do ATER. A seguir, são descritos os principais elementos dessa arquitetura.



- **CORE**
    - **CORE Web**: este módulo de software é instanciado no CORE e consiste em um sistema Web através do qual clientes podem criar regras para identificação do perfil de tráfego, visualizar suas regras criadas e consultar estatísticas coletadas sobre estas regras. Este módulo é também acessado pelos administradores, que utilizam o sistema para gerenciar usuários, manter regras, monitorar estatísticas de tráfego, controlar remotamente os RACES e configurar/manter o próprio serviço. Este módulo foi desenvolvido em PHP.

    - **ATER DB**: esta é a base de dados do sistema, hospedada no CORE, onde estão armazenados os dados dos usuários e seus privilégios, os dados das regras definidas pelos usuários, os dados dos circuitos criados para atender os diferentes perfis de tráfego, as estatísticas coletadas pelo serviço, os dados relacionados aos RACEs (localização, conexões, estado, etc), além de outros dados de configuração do próprio serviço. Esta base de dados é mantida no PostgreSQL.

    - **CORE Main**: este módulo é o núcleo do serviço de gerenciamento de regras instanciado no CORE, sendo formado por dois componentes principais:

        - **Circuit Manager**: responsável por coordenar os RACES nas atividades de aplicação de regras definidas pelo usuário e estabelecimento da conectividade entre as extremidades dos circuitos dinâmicos.

        - **RACE Checker**: é um daemon que verifica, periodicamente, o estado dos RACES e atualiza a base de dados com a informação mais recente.

    - **CORE OSCARS Driver**: camada de software instanciada no CORE, responsável pela integração com o SE-CIPÓ. Essa camada é implementada na linguagem Java e funciona como um driver ou uma ponte de comunicação entre o CORE Main, desenvolvido em PHP, e o serviço de reserva de circuito dinâmico do OSCARS, implementado em Java.

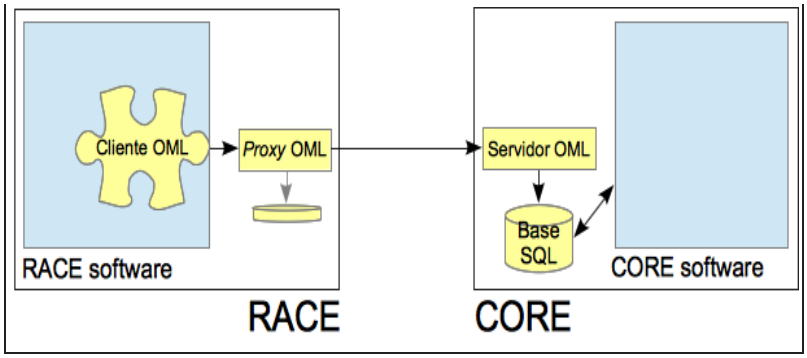
    - **CORE Comm**: camada de software que implementa a comunicação entre o CORE e os RACEs. A base da comunicação entre CORE e RACES segue um modelo de execução remota sobre um canal seguro. Assim, a comunicação no sentido CORE para RACE é implementada usando um wrapper da biblioteca libssh2 para clientes PHP.

- **RACE**
    - **RACE Daemon**: a parte inicial da comunicação através de circuitos dinâmicos consiste em interceptar o tráfego que não deve seguir através da rede de pacotes convencional e reinjetá-lo no caminho definido pelo circuito dinâmico. O RACE Daemon é um módulo de software instanciado no RACE, responsável pela lógica de filtragem e encaminhamento transparente desses pacotes. Para interceptação do tráfego, é utilizada uma solução nativa do núcleo do sistema operacional Linux, conhecida como Netfilter, em conjunto com Linux bridge. Para a reinjeção dos pacotes dentro do circuito dinâmico e sua correspondente remoção na outra extremidade do circuito, é usada a API de raw sockets, em linguagem C.

    - **CORE Checker**: é um daemon que verifica, periodicamente, o estado da conexão com o CORE e comunica o RACE Daemon caso haja algum problema com essa conexão.

    - **RACE Comm**: camada de software que implementa a comunicação entre o RACE e o CORE. A base da comunicação entre CORE e RACES segue um modelo de execução remota sobre um canal seguro. Assim, a comunicação no sentido RACE para CORE é implementada usando libssh2.

Além dos módulos de software descritos acima, o ATER é composto também por um módulo de coleta de dados estatísticos. A coleta das estatísticas é realizada onde o tráfego flui, ou seja, nos RACEs e, então, encaminhada para o ponto central de operação do serviço, ou seja, para o CORE. A partir do CORE, as estatísticas podem ser consultadas e analisadas. Para implementar a coleta distribuída e a persistência centralizada das estatísticas, é usada a ferramenta **OML**. A figura abaixo ilustra o uso dessa ferramenta no contexto do ATER.



Tipos de Regras

No ATER, existem dois tipos de regras:

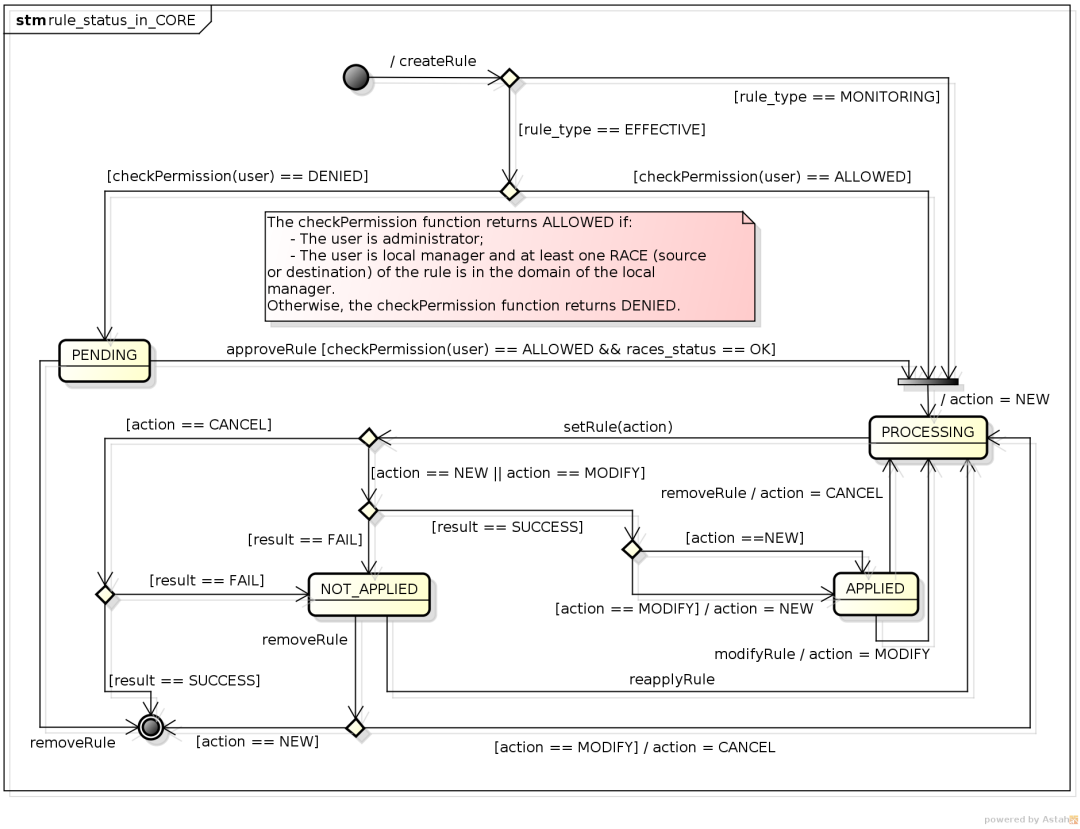
- **Efetiva**: Permite solicitar um circuito dinâmico quando um tráfego que casa com ela é identificado.
- **Monitoramento**: Não permite solicitar circuito quando um tráfego que casa com ela é identificado. Entretanto, todo tráfego que casa com a regra é contabilizado.

Independente do tipo da regra, todo o tráfego que casa com ela é contabilizado.

Além de criar e remover uma regra, ainda é possível modificá-la. Entretanto, atualmente o serviço só permite a modificação do tipo da regra, de monitoramento para efetiva.

Durante seu ciclo de vida, uma regra pode passar por diferentes estados, dependendo do seu tipo e do componente onde a regra está instanciada. Os componentes que trabalham com as regras atualmente são o CORE e o RACE.

1. **Regra no CORE**



Pode-se dividir as transições da regra no CORE entre os processos de criação, aplicação e remoção da regra.

O processo de criação da regra contém alguns detalhes que dependem do tipo da regra e do usuário que está criando a regra.

- **Regra Efetiva**
  - ***Usuário Administrador ou Gerente Local***
    Quando uma regra efetiva é criada por um Gerente Local (pelo menos um dos RACEs da regra deve estar sobre o domínio dele), ou por um Administrador, ela é considerada aprovada automaticamente.
  - ***Cliente***
    Quando uma regra efetiva é criada por um usuário que não tem permissões de aprovação direta, como um cliente, a regra automaticamente assume o estado "pendente" (**PENDING**), indicando que ela deverá aguardar a aprovação do Gerente Local, dependendo da configuração da regra, ou de um Administrador.

- **Regra de Monitoramento**
  Quando uma regra de monitoramento é criada, independente do tipo do usuário que a criou, ela é considerada aprovada automaticamente.

Quando uma regra é considerada aprovada, imediatamente ela assume o estado "processando" (**PROCESSING**) e o processo para aplicá-la nos RACEs é iniciado.

O processo de aplicação tem dois resultados possíveis:

- **SUCESSO**: Se o processo de aplicação da regra obtiver sucesso, ou seja, a regra for aplicada com sucesso em ambos os RACEs, a regra assume o estado "aplicada" (**APPLIED**).
- **FALHA**: Se o processo de aplicação da regra falha, ou seja, a regra não pode ser aplicada em um ou mais RACEs, a regra assume o estado "não-aplicada" (**NOT_APPLIED**).

Uma regra pode ser removida quando ela está em qualquer estado, exceto "processando". Nesse processo, algumas verificações acontecem sobre a regra para determinar qual o procedimento correto para sua exclusão.

- **Exclusão de regra que nunca foi aplicada:**

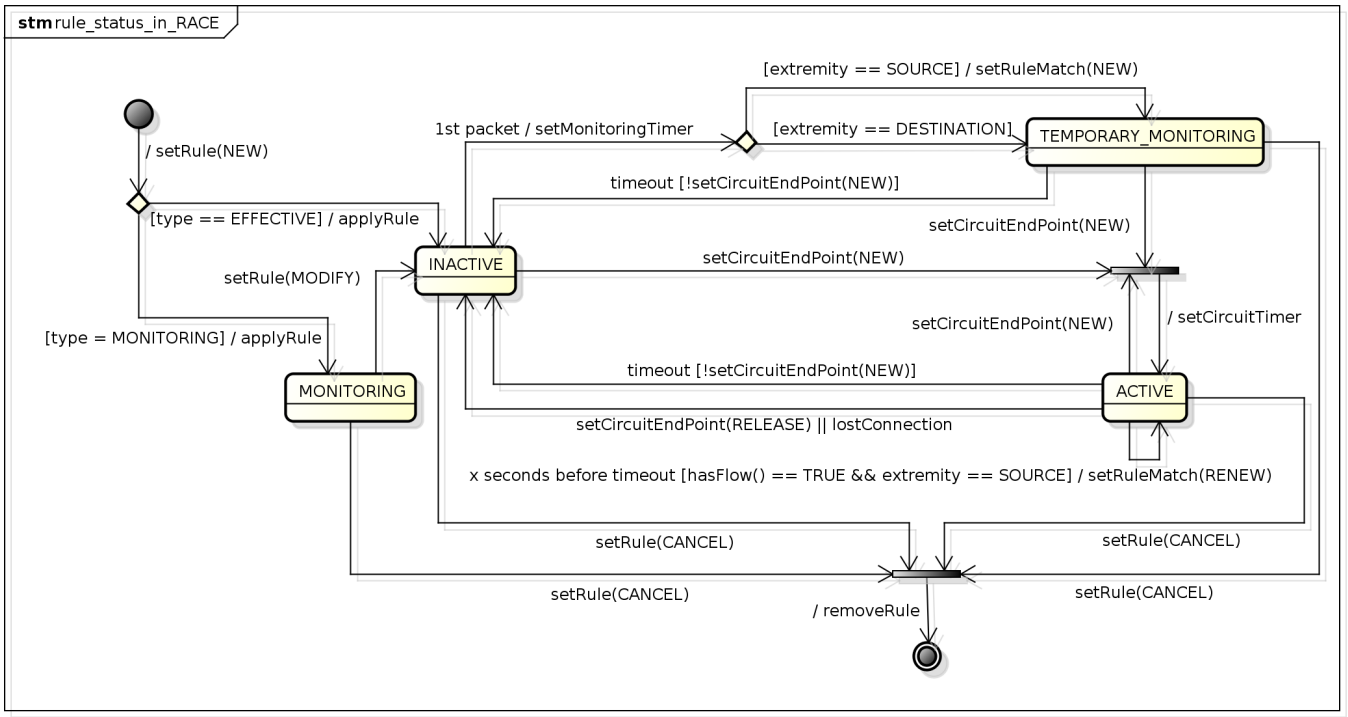
Quando o usuário solicita a exclusão de uma regra que foi criada e nunca foi aplicada com sucesso nos RACEs, seja por não aprovação ou por algum problema durante a aplicação nos RACEs, a regra é excluída diretamente e ciclo de vida dela termina, sem que os RACEs precisem ser contatados.

- **Exclusão de regras que já foram aplicas:**
  Quando o usuário solicita a exclusão de uma regra que foi criada e já foi aplicada pelo menos uma vez nos RACEs, a regra assume o estado "processando" e os RACEs são contatados para que também excluam a regra. Caso a exclusão obtenha sucesso, a regra é excluída do CORE e o ciclo de vida dela termina. Caso a exclusão não obtenha sucesso, o próprio CORE vai, automaticamente, tentar excluir a regra em outro momento. Neste último caso, o usuário não pode mais tentar excluir a regra.

Quando o usuário solicita a modificação uma regra, a regra assume o estado "processando" e então a modificação é enviada para os RACEs. Caso essa modificação aconteça com sucesso nos RACEs, a regra assume o estado "aplicada" e ação "NEW". Caso contrário, a regra assume o estado "não-aplicada". Entretanto, uma regra só pode ser modificada se ela já tiver sido aplicada, ou seja, quando ela está no estado "aplicada".

Periodicamente, um componente do serviço tenta reaplicar regras cujas aplicações anteriores foram mal sucedidas. As tentativas são repetidas até que a regra seja aplicada com sucesso. Essa reaplicação acontece com base na ação que foi mal sucedida para a regra, ou seja, a reaplicação pode acontecer para uma regra não aplicada, não modificada ou não removida nos RACEs.

2. **Regra no RACE**



No RACE as transições de estado da regra estão condicionadas ao tipo dela.

- **Regra Efetiva**
  Quando o RACE recebe uma regra efetiva, ela assume o estado "inativa" (**INACTIVE**), que é o estado inicial da regra, de onde ela pode mudar de estado com base nos eventos que acontecem.

  Quando a regra está no estado "inativa", dois eventos podem ocasionar a mudança de estado da regra:
  - **Tráfego que casa com a regra é identificado**
    Neste caso, quando o RACE identifica algum tráfego que casa com as configurações da regra, ela assume o estado "monitoramento-temporário" (**TEMPORARY_MONITORING**) e, se o RACE for o de origem da regra, uma mensagem *setRuleMatch(NEW)* é enviada para o CORE. Além disso, um temporizador é disparado para que o RACE possa voltar a regra para o estado "inativa", caso uma mensagem *setCircuitEndPoint(NEW)* não seja recebida em um intervalo de tempo.

  - **Uma mensagem *setCircuitEndPoint(NEW)* é recebida**
    Quando o RACE recebe uma mensagem *setCircuitEndPoint(NEW)* com as informações de um circuito para uma determinada regra, essa regra assume o estado "ativa" (**ACTIVE**).

  Quando a regra está no estado "monitoramento-temporário" e o RACE recebe uma mensagem *setCircuitEndPoint(NEW)* com as informações de um circuito para uma determinada regra, essa regra assume o estado "ativa".

  Quando a regra assume o estado "ativa", um temporizador é disparado para que o RACE possa controlar as seguintes operações:
  - **Solicitação de renovação de circuito**
    Caso o tempo para expiração do circuito recebido esteja próximo do fim e ainda haja tráfego casando com a regra, o RACE solicitará a renovação do circuito para o CORE, através do envio de uma mensagem *setRuleMatch(RENEW)*.

  - **Voltar a regra para o estado "inativa"**
    Caso o RACE não receba uma mensagem *setCircuitEndPoint(NEW)*, mesmo depois de enviar uma mensagem *setRuleMatch(RENEW)* para o CORE, a regra então assume o estado "inativa" novamente.

  Caso o RACE receba uma mensagem *setCircuitEndPoint(NEW)*, um novo temporizador é disparado e a regra permanece no estado "ativa".

  Além disso, a regra pode sair do estado "ativa" para "inativa" caso o RACE receba uma mensagem *setCircuitEndPoint(RELEASE)*, que significa que o RACE deve deixar de transmitir dados pelo circuito atual, e também caso o RACE identifique que perdeu a conexão com o CORE.

- **Regra de Monitoramento**
  Quando o RACE recebe uma regra de monitoramento, ela assume o estado "monitoramento" (**MONITORING**). Nesse estado, todas as estatísticas são colhidas, mas caso algum tráfego case com a regra, nenhuma mudança de estado acontece.

  Quando o RACE recebe uma solicitação de modificação da regra, a regra assume o estado "inativa" (**INACTIVE**). A partir desse ponto, a regra pode assumir as mesmas transições descritas no item acima, para regra efetiva.

Independentemente do estado que uma regra se encontra, caso o RACE receba uma solicitação de exclusão da regra, a regra é removida do RACE e o ciclo de vida dela termina.