Simulador para um Modelo de Busca Oportunística de Conteúdos em Redes Orientadas a Informação

Desenvolvido pelo Laboratório de Modelagem, Análise e Desenvolvimento de Redes e Sistemas Computacionais

>*Flow do Programa*

Ao ser iniciado, o programa começa na função main() descrita em ICNSimulator.cpp, nela lendo o arquivo de configurações SimConfig.txt, iniciando então a simulação. O controle é passado para OperationManager.cpp, onde todas as variáveis de simulação são alocadas e inicializadas, incluindo o tempo de simulação, domínios e conteúdos, cada nó de todos os tipos possíveis. Feito isso, ele linka os nós de mesmo domínio entre si e, quando necessário, ao de um domínio acima, especificado na configuração.

Após inicializar todos os nós devidamente, a simulação é iniciada e o controle passa para a função *ThreadEvHandler*, onde o loop abaixo é repetido até o fim da simulação:

- Se a lista de eventos está vazia - nada a ser feito
- Caso contrário, chamar a função dequeue()
- Avançar o tempo de simulação para o tempo do evento
- Chamar a rotina de processamento do evento
- Uma vez processado, apagamos o evento tratado

Os eventos da lista são processados dentro da classe do nó que as criou, ou seja, esse processamento é feito através de funções membro dos nós, não na função *ThreadEvHandler*. A função dequeue retira da lista o elemento de menor tempo, ou seja, aquele cujo "*timestamp*" representa o menor tempo de simulação.

Uma vez terminada a simulação, é chamada a rotina de interrupção de nós, que desabilita cada nó individualmente, registrando no log as informações relevantes pós-simulação. Uma vez escrito o log, a memória é desalocada e a execução termina. Os resultados da simulação podem então ser tirados do log, para que sejam tratados e analisados.

>Tipos de Nós e seus Eventos

- **Base Node:** Nó básico do qual todos os demais herdam. Consiste em um buffer próprio para armazenamento de conteúdo, uma lista de nós do domínio acima e outra do domínio concorrente, para que possa mandar requisições dentro do próprio domínio, ou para um domínio acima. Conta com métodos de Processamento de Transmissão e Recepção de requisições, que podem ser sobrescritos pelos nós que herdam de Base Node.

- **Client Node:** Nó que representa os clientes na rede, herdando de Base Node. Responsável por gerar e transmitir requisições por conteúdo pela rede. Sendo assim, é o responsável pela criação do novo pacote de requisição, colocando na lista de eventos a requisição pelo conteúdo desejado, a ser tratado pelo nó de domínio acima, tipicamente um Router Node.

- **Router Base Node:** Nó base de um roteador, ou seja, representa todo e qualquer roteador conectado à rede, herdando de Base Node. A maneira como o roteador trata os pedidos depende da sua implementação, podendo ser definida através de probabilidade fixa, sem fila de entrada ou com fila de entrada. Assim, temos três classes para o roteador: Router Prob Node, ou seja, a probabilidade do conteúdo estar no cache é dada por um valor fixo. Router Rc Node NQueue, ou seja, todo pacote recebido é atendido imediatamente, usando de um contador RC para verificar se o conteúdo está ou não no cache. Router Rc Node Queue, o pacote recebido é colocado numa fila de espera. Esses nós são responsáveis por receber as requisições e trata-las ou respondendo ela com um cache hit ou redirecionando o pedido para outro roteador (ou, dependendo do caso, para o servidor).

- **Server Node:** Nó do servidor distribuidor de conteúdo, responsável por atender toda e qualquer requisição que não conseguiu ser atendida nos domínios, ou seja, o conteúdo não estava disponível em nenhum cache da rede. Esse nó somente trata requisições de conteúdo devolvendo tal conteúdo, nunca irá lançar uma requisição a outro nó, até porque ocupa o nível mais alto.