O ns-3 é um simulador de redes de eventos discretos voltado principalmente para pesquisa e uso educacional, o projeto com código aberto que desenvolve o ns-3 foi iniciado em 2006. O ns-3 fornece modelos de como as redes de dados de pacote funcionam e fornece uma simulação para os usuários conduzirem simulação de experimentos. Algumas das razões para usar o ns-3 incluem a realização de estudos que são mais difíceis ou impossíveis de realizar com sistemas reais, para estudar o comportamento do sistema é simulado em um ambiente altamente controlado. Os usuários notarão que o modelo disponível definido no ns-3 se concentra na modelagem de como os protocolos e redes da Internet funcionam, mas o ns-3 não se limita aos sistemas da Internet, os usuários também estão usando para modelar em sistemas não baseados na internet.

O ns-3 foi projetado como um conjunto de bibliotecas que podem ser combinadas entre si e também com outras bibliotecas de software externas. Embora algumas plataformas de simulação forneçam aos usuários um único ambiente de interface gráfica do usuário integrado em que todas as tarefas são realizadas, o ns-3 é mais modular a esse respeito. Vários animadores externos e análise de dados e as ferramentas de visualização podem ser usadas com o ns-3. No entanto, os usuários devem esperar trabalhar na linha de comando e com ferramentas de desenvolvimento de software C ++ e / ou Python.

O ns-3 é usado principalmente em sistemas Linux ou macOS, embora exista suporte para sistemas BSD e também para estruturas do Windows que podem construir código Linux, como subsistema Windows para Linux ou Cygwin. Nativo o Windows Visual Studio não tem suporte no momento, embora um desenvolvedor esteja trabalhando em suporte futuro.

O ns-3 não é um produto de software com suporte oficial de nenhuma empresa. O suporte para ns-3 é feito com o melhor esforço com base no fórum ns-3-users.

Existem vários recursos importantes dos quais qualquer usuário do ns-3 deve estar ciente. O site principal https://www.nsnam.org que fornece acesso a informações básicas sobre o sistema ns-3. O código-fonte pode ser encontrado e navegado em GitLab.com: https://gitlab.com/nsnam/, que contém a árvore de desenvolvimento no repositório denominado ns-3-dev.

Os sistemas de software complexos precisam de alguma forma de gerenciar a organização e as mudanças no código e documentos subjacentes. Até recentemente, o projeto ns-3 usava Mercurial como seu sistema de gerenciamento de código-fonte, mas em dezembro 2018, mude para o Git.

Depois de fazer o download do código-fonte em seu sistema local, você precisará compilar essa fonte para produzir programas. Assim como no caso do gerenciamento de código-fonte, existem muitas ferramentas disponíveis para realizar esta função. Provavelmente, a mais conhecida dessas ferramentas é o make . Além de ser o mais conhecido, make é rovavelmente o mais difícil de usar em um sistema muito grande e altamente configurável. Por causa disso, muitas alternativas foram desenvolvidas recentemente usando a linguagem Python.

O sistema de construção Waf é usado no projeto ns-3. É um da nova geração de sistemas de compilação baseados em Python. Conforme mencionado anteriormente, o script no ns-3 é feito em C ++ ou Python. A maior parte da API ns-3 está disponível em Python, mas os modelos são escritos em C ++ em ambos os casos.

O ns-3 é construído como um sistema de bibliotecas de software que funcionam juntas. Os programas do usuário podem ser escritos vinculados ou importados das bibliotecas.

O ns-3 é distribuído como código-fonte, o que significa que o sistema de destino precisa ter um ambiente de desenvolvimento de software para construir as bibliotecas primeiro e, em

seguida, construir o programa do usuário. ns-3 poderia, em princípio, ser distribuído como bibliotecas pré-construídas para sistemas selecionados, e no futuro pode ser distribuído dessa forma, mas no momento, muitos usuários realmente fazem seu trabalho editando o próprio ns-3, portanto, ter o código-fonte disponível para reconstruir as bibliotecas é útil.

Três maneiras de baixar e construir o ns-3. O primeiro é baixar e construir um lançamento oficial do site principal. A segunda é buscar e construir cópias de desenvolvimento de uma instalação básica do ns-3. A terceira é usar uma ferramenta de construção adicional para baixar mais extensões para o ns-3. Vamos examinar cada um desde o as ferramentas envolvidas são ligeiramente diferentes. Para a maioria dos usuários do ns-3, as permissões de root não são necessárias e o uso de uma conta de usuário sem privilégios é recomendado.

Todo o conjunto de bibliotecas ns-3 disponíveis tem várias dependências de bibliotecas de terceiros, mas a maioria do ns-3 pode ser construído e usado com suporte para alguns componentes comuns geralmente instalados por padrão: um compilador C ++, uma instalação

de Python, um editor de código-fonte como vim, emacs ou Eclipse e, se estiver usando os repositórios de desenvolvimento, uma instalação do sistema de controle de código-fonte Git. A maioria dos usuários iniciantes não precisa se preocupar se sua configuração relata alguns recursos opcionais ausentes do ns-3, mas para aqueles que desejam uma instalação completa, o projeto fornece um wiki que inclui páginas com muitas dicas e sugestões úteis.

No jargão da Internet, um dispositivo de computação que se conecta a uma rede é chamado de host ou, às vezes, de sistema final. Porque ns-3 é um simulador de rede, não especificamente um simulador de Internet, intencionalmente não usamos o termo host, pois ela está intimamente associada com a Internet e seus protocolos. Em vez disso, usamos um termo mais genérico também usado por outra simuladores que se originam na Teoria de Grafos - o nó.

No ns-3, a abstração básica do dispositivo de computação é chamada de nó. Esta abstração é representada em C ++ pela classe node. A classe node fornece métodos para gerenciar as representações de dispositivos de computação em simulações. Você deve pensar em um Nó como um computador ao qual adicionará funcionalidade. Um adiciona coisas como aplicativos, pilhas de protocolo e placas periféricas com seus drivers associados para permitir que o computador faça um trabalho útil. Nós usamos o mesmo modelo básico no ns-3.

Normalmente, o software de computador é dividido em duas classes amplas. O software do sistema organiza vários recursos de computador tais como a memória, ciclos de processador, disco, rede, etc., de acordo com algum modelo de computação. software de sistema normalmente não usa esses recursos para concluir tarefas que beneficiam diretamente um usuário. Um usuário normalmente executaria um aplicativo que adquire e usa os recursos controlados pelo software do sistema para cumprir algum objetivo. Frequentemente, a linha de separação entre o software do sistema e do aplicativo é feita na mudança de nível de privilégio que acontece em armadilhas do sistema operacional. No ns-3 não existe um conceito real de sistema operacional e especialmente nenhum conceito de privilégio níveis ou chamadas de sistema. No entanto, temos a ideia de um aplicativo. Assim como os aplicativos de software são executados em computadores para realizar tarefas no “mundo real”, os aplicativos do ns-3 são executados nos nós do ns-3 para conduzir simulações no mundo simulado.

No ns-3, a abstração básica para um programa de usuário que gera alguma atividade a ser simulada é o aplicativo. Esse a abstração é representada em C ++ pela classe Application.

A classe Application fornece métodos para gerenciar as representações de nossa versão de aplicativos de nível de usuário em simulações. Espera-se que os desenvolvedores se

especializem na aplicação de classe no sentido de programação orientada a objeto para criar novas aplicações.

No mundo real, pode-se ligar um computador a uma rede. Muitas vezes, os meios de comunicação sobre os quais fluxos de dados nessas redes são chamados de canais. Ao conectar o cabo Ethernet ao plugue na parede, você está conectando o computador para um canal de comunicação Ethernet. No mundo simulado do ns-3, conecta-se um Nó a um objeto que representa um canal de comunicação. Aqui, a abstração básica da sub-rede de comunicação é chamada de canal e é representada em C ++ pela classe Channel.

A classe Channel fornece métodos para gerenciar objetos de sub-rede de comunicação e conectar nós a eles. Os canais também podem ser especializados por desenvolvedores no sentido de programação orientada a objetos. Uma especialização de canal pode modelar algo tão simples como um fio. O canal especializado também pode modelar coisas tão complicadas quanto um grande Switch Ethernet, ou espaço tridimensional cheio de obstruções no caso de redes sem fio.

Antes, se você quisesse conectar um computador a uma rede, teria que comprar um tipo específico de rede cabo e um dispositivo de hardware chamado na terminologia de PC uma placa periférica que precisava ser instalada em seu computador. Se a placa periférica implementou alguma função de rede, elas foram chamadas de placas de interface de rede ou NICs. Hoje, a maioria dos computadores vem com o hardware de interface de rede integrado e os usuários não veem esses blocos de construção. Uma NIC não funcionará sem um driver de software para controlar o hardware. No Unix e Linux, um pedaço de disco periférico ware é classificado como um dispositivo. Os dispositivos são controlados usando drivers de dispositivo, e os dispositivos de rede NICs são controlados usando drivers de dispositivo de rede conhecidos coletivamente como dispositivos de rede. No Unix e Linux, você se refere a esses dispositivos de rede por nomes como eth0.
No ns-3, a abstração do dispositivo de rede cobre tanto o driver de software quanto o hardware simulado. Um dispositivo de rede está “dentro stalled” em um Node para permitir que o Node se comunique com outros Nodes na simulação por meio de canais. Somente como em um computador real, um Node pode ser conectado a mais de um Canal por meio de múltiplos NetDevices.

Em uma rede real, você encontrará computadores host com placas de rede adicionadas ou integradas. No ns-3, diríamos que você encontrará Nós com NetDevices anexados. Em uma grande rede simulada, você precisará organizar muitas conexões entre Nós, dispositivos de rede e canais. Uma vez que conectar NetDevices a nós, NetDevices a canais, atribuir endereços IP, etc., são tão comuns tarefas no ns-3, fornecemos o que chamamos de ajudantes de topologia para tornar isso o mais fácil possível. Por exemplo, pode demorar muitas operações distintas do núcleo do ns-3 para criar um NetDevice, adicionar um endereço MAC, instalar esse dispositivo de rede em um Nó, configurar a pilha de protocolo do nó e, em seguida, conectar o NetDevice a um canal. Ainda mais operações seriam necessárias para conectar vários dispositivos em canais multiponto e, em seguida, conectar redes individuais em internetworks. Nós fornecemos objetos auxiliares de topologia que combinam essas muitas operações distintas em um fácil de usar modelo para sua conveniência.

Muitos sistemas grandes suportam algum tipo de recurso de registro de mensagens e o ns-3 não é uma exceção. Em alguns casos, apenas as mensagens de erro são registradas no “console do operador” que normalmente é stderr em sistemas baseados em Unix. Em outro sistemas, mensagens de aviso podem ser emitidas, bem como mensagens informativas mais detalhadas. Em alguns casos, o registro as facilidades são usadas para enviar mensagens de depuração que podem rapidamente transformar a saída em um borrão.

O ns-3 considera que todos esses níveis de detalhamento são úteis e oferecemos uma abordagem selecionável de vários níveis ao registro de mensagens. O registro pode ser desabilitado completamente, habilitado componente por componente ou habilitado globalmente; e fornece níveis de detalhamento selecionáveis. O módulo de registro do ns-3 fornece um método simples e relativamente fácil para usar a maneira de obter informações úteis de sua simulação. O registro deve ser preferencial para informações de depuração, avisos, mensagens de erro ou a qualquer momento você deseja obter facilmente uma mensagem rápida de seus scripts ou modelos.

Você pode adicionar um novo registro às suas simulações fazendo chamadas para o componente de registro por meio de várias macros. Vamos fazer isso em o script myfirst.cc que temos no diretório de rascunho. Outra maneira de alterar o comportamento dos scripts do ns-3 sem edição e construção é por meio de argumentos de linha de comando. Nós fornecemos um mecanismo para analisar os argumentos da linha de comando e definir automaticamente as variáveis locais e globais com base em esses argumentos. Todo o objetivo da simulação é gerar saída para um estudo mais aprofundado, e o sistema de rastreamento do ns-3 é um mecanismo primário para isso. Uma vez que o ns-3 é um programa C ++, os recursos padrão para gerar saída de programas C ++ podem ser usados.

Você pode até usar o módulo de registro para adicionar uma pequena estrutura à sua solução. Existem muitos problemas conhecidos gerado por tais abordagens e, portanto, fornecemos um subsistema de rastreamento de eventos genérico para resolver os problemas que pensamento eram importantes.

Os objetivos básicos do sistema de rastreamento ns-3 são: Para tarefas básicas, o sistema de rastreamento deve permitir que o usuário gere rastreamento padrão para fontes de rastreamento populares, e personalizar quais objetos geram o rastreamento; os usuários intermediários devem ser capazes de estender o sistema de rastreamento para modificar o formato de saída gerado ou inserir novas fontes de rastreamento, sem modificar o núcleo do simulador; os usuários avançados podem modificar o núcleo do simulador para adicionar novas fontes de rastreamento e sumidouros.

O sistema de rastreamento ns-3 é construído sobre os conceitos de fontes de rastreamento independentes e coletores de rastreamento, e um sistema uniforme mecanismo para conectar fontes a sumidouros. Fontes de rastreamento são entidades que podem sinalizar eventos que acontecem em uma simulação e fornecer acesso a dados subjacentes interessantes. Por exemplo, uma fonte de rastreamento pode indicar quando um pacote é recebido por um dispositivo de rede e fornecer acesso ao conteúdo do pacote para coletores de rastreamento interessados. As fontes de rastreamento não são úteis por si mesmas, elas devem estar "conectadas" a outras partes do código que realmente fazem algo útil com as informações fornecidas pelo coletor. Os coletores de rastreamento são consumidores dos eventos e dados fornecidos pelo fontes de rastreamento. Por exemplo, pode-se criar um coletor de rastreamento que quando conectado à fonte de rastreamento do anterior exemplo imprima partes interessantes do pacote recebido. A justificativa para essa divisão explícita é permitir que os usuários anexem novos tipos de coletores às fontes de rastreamento existentes, sem exigindo edição e recompilação do núcleo do simulador. Assim, no exemplo acima, um usuário poderia definir um novo coletor de rastreamento em seu script e anexá-lo a uma fonte de rastreamento existente definida no núcleo de simulação editando apenas o script do usuário. Neste tutorial, vamos percorrer algumas fontes e coletores predefinidos e mostrar como eles podem ser personalizados com pouco esforço do usuário. Consulte o manual do ns-3 ou as seções de procedimentos para obter informações sobre a configuração de rastreamento

avançado, incluindo estendendo o namespace de rastreamento e criando novas fontes de rastreamento.

O ns-3 fornece funcionalidade auxiliar que envolve o sistema de rastreamento de baixo nível para ajudá-lo com os detalhes envolvidos em configurar alguns rastreamentos de pacotes de fácil compreensão. Se você habilitar esta funcionalidade, verá a saída em arquivos ASCII daí o nome. Para aqueles familiarizados com a saída do ns-2, este tipo de traço é análogo ao out.tr gerado por muitos scripts.

Assim como você já viu muitas vezes antes, você verá algumas mensagens do Waf e, em seguida, “'build' concluído com sucesso” com algum número de mensagens do programa em execução. Quando executado, o programa terá criado um arquivo chamado myfirst.tr . Por causa da maneira como o Waf funciona, o arquivo não é criado no diretório local, ele é criado no diretório de nível superior do repositório por padrão. Se você quiser controle onde os traços são salvos você pode usar o --cwd opção de Waf para especificar isso.