Disciplina de Simulação e Análise de Desempenho de Redes de Computadores – Autores: Alunos da turma SADRC0001 2015

1) Como passar o número do nó via CallBack?

Código padrão:	static void CwndChange (uint32_t oldCwnd, uint32_t newCwnd) {
	  NS_LOG_UNCOND ("cwnd\t" << Simulator::Now ().GetSeconds () << "\t" << newCwnd);
	}
	…
	
	tcpSocket->TraceConnectWithoutContext("CongestionWindow", MakeCallback (&CwndChange));
	…

Solução:	static void CwndChange (**uint16_t numNode**, uint32_t oldCwnd, uint32_t newCwnd) {
	  NS_LOG_UNCOND ("cwnd n" << numNode << "\t" << Simulator::Now ().GetSeconds () << "\t" << newCwnd);
	}
	…
	tcpSocket->TraceConnectWithoutContext("CongestionWindow", Make**Bound**Callback (&CwndChange,**numNode**));
	…

2) Como calcular a quantidade de bytes enviados efetivamente via TCP?

Explicação:	Como calcular a quantidade de bytes enviados efetivamente via TCP? Pode-se dizer taxa de transmissão efetiva em bytes, ou simplesmente throughput. Para isso é possível utilizar o trace “NextTxSequence” que carrega o valor de bytes que foram recebidos até o momento.

Solução:	static void NextTxSequence (uint16_t numNode, SequenceNumber32 oldSequence, SequenceNumber32 newSequence) {
	  NS_LOG_UNCOND ("nxts n" << numNode << "\t" << Simulator::Now ().GetSeconds () << "\t" << newSequence.GetValue());
	}
	…
	tcpSocket->TraceConnectWithoutContext ("**NextTxSequence**", MakeBoundCallback (&NextTxSequence,numNode));
	…

Dicas:	Para calcular a taxa em Mbps, basta acumular a diferença entre os valores de “NextTxSequence” dentro de um segundo e converter de bytes para bits.
Problemas:	Se finalizar a simulação interrompendo abruptamente a aplicação cliente e a aplicação servidora, serão perdidas as informações sobre “NextTxSequence” e isso pode refletir nos resultados obtidos.

3) Como calcular a quantidade de bytes que sairam de um nó?

Explicação:	É possível calcular a quantidade de bytes que foram enviados pelo host, mas não necessariamente aqueles que chegaram com sucesso ao destino, permitindo, assim, fazer uma comparação com os resultados de taxa efetiva (conforme item 2). Pode-se utilizar o trace “OutputBytes” de aplicações BulkSendApplication ou OnOffAplication (conforme item 4); ou, neste caso, inspecionar o pacote que foi enviado através do trade “SendOutGoing”.

Solução:	static void OutputBytes (uint16_t numNode, const Ipv4Header &iph, Ptr <const Packet> pkt, uint32_t iface) {
	  NS_LOG_UNCOND ("outb n" << numNode << "\t" << Simulator::Now ().GetSeconds () << "\t" << **pkt->GetSize()**);
	}
	…
	std::stringstream stdCBack;
	stdCBack.str("/$ns3::NodeListPriv/NodeList/5/$ns3::Node/$ns3::Ipv4L3Protocol/**SendOutgoing**");
	Config::ConnectWithoutContext(stdCBack.str(), MakeBoundCallback(&OutputBytes,5));
	…

Dica:	O valor de pkt-GetSize() irá fornecer o tamanho do pacote que chegou até a camada 3, para um calculo mais próximo do que se trata de TCP ou UDP, pode-se subtrair a quantidade de bytes do cabeçalho desses protocolos.

4) Como configurar aplicações diferentes de MyAPP?

Explicação:	A aplicação MyApp é uma classe de exemplo do NS3, porém esta não possui o trace “OutputBytes”. Para isso é possível utilizar as classes BulkSendApplication para TCP e OnOffApplication para UDP. Abaixo um exemplo com BulkSendApplication que dispensa o uso do MyAPP.

Solução:	Ptr<PacketProbe> packetProbe = CreateObject<PacketProbe>();
	packetProbe->Enable();
	BulkSendHelper app ("ns3::TcpSocketFactory", InetSocketAddress (csmaInterfaces.GetAddress (3), sinkPort));
	app.SetAttribute ("MaxBytes", UintegerValue (0));
	ApplicationContainer sourceApp = app.Install (wifiStaNodes.Get (0));
	sourceApp.Start (Seconds (1.0));
	sourceApp.Stop (Seconds (40.0));
	packetProbe->ConnectByPath("/$ns3::NodeListPriv/NodeList/5/$ns3::Node/ApplicationList/0/$ns3::**BulkSendApplication/Tx**");
	packetProbe->TraceConnectWithoutContext(**"OutputBytes"**, MakeBoundCallback (&OutputBytes,5));
	…

Problemas:	Ao se usar a aplicação com BulkSendApplication é possível capturar o trace “OutputBytes”, porém até o momento dessa anotação não foi encontrado como fazer o trace “CongestionWindow” e o trace “NextTxSequence”.

5) Como saber os paths carregados durante a simulação?

Explicação:	Conforme item 4, o trace BulkSendApplication/Tx poderia ser identificado via ConfigStore após uma simulação de NS3. O ConfigStore irá listar todos os paths carregados na simulação. Abaixo como isso pode ser feito.
Solução:	… (Antes de carregar os nós) ConfigStore config; config.ConfigureDefaults (); … … (Depois de carregar os nós e as aplicações) config.ConfigureAttributes (); … Exemplo de linha de comando: $ time ./waf --run "scratch/scriptns3 --ns3::ConfigStore::Mode=Save --ns3::ConfigStore::Filename=**config.txt**" … Verificar o arquivo config.txt.

6) Como capturar a quantidade de colisões que ocorrem em uma rede WiFi?

Explicação:	Até o momento desta anotação não foi encontrada uma maneira de se capturar uma colisão efetivamente, porém é possível que os seguintes traces se tratem de colisões que ocorreram em uma simulação: MacTxDrop, PhyRxDrop e PhyTxDrop.

Solução:	static void PossibleCollision (Ptr<const Packet> p) {
	  NS_LOG_UNCOND ("pcol at\t" << Simulator::Now ().GetSeconds ());
	}
	…
	std::stringstream stdCBack;
	stdCBack.str("/NodeList/0/DeviceList/1/$ns3::WifiNetDevice/Mac/**MacTxDrop**");
	Config::ConnectWithoutContext(stdCBack.str(), MakeCallback(&PossibleCollision));
	stdCBack.str("/NodeList/0/DeviceList/1/$ns3::WifiNetDevice/Phy/**PhyRxDrop**");
	Config::ConnectWithoutContext(stdCBack.str(), MakeCallback(&PossibleCollision));
	stdCBack.str("/NodeList/0/DeviceList/1/$ns3::WifiNetDevice/Phy/**PhyTxDrop**");
	Config::ConnectWithoutContext(stdCBack.str(), MakeCallback(&PossibleCollision));
	…

Observações:	Até o momento dessa anotação, durante algumas simulações somente o trace PhyRxDrop retornou informações.

7) Como configurar uma simulação para diversos nós através de um parâmetro N?

Explicação:	Os exemplos do NS3 já ensinam como adicionar múltiplos devices, porém não ensinam como colocar diversos nós e suas aplicações.

Solução (exemplo):	…
	uint32_t nWifi = 5;
	uint32_t nwStart[] = { 0,10,20,30,40};
	uint32_t nwStop[] = {50,50,50,50,50};
	…
	for (uint16_t **numNode** = 1; numNode < nWifi; numNode = numNode + 1) {
	Ptr<Socket> tcpSocket = Socket::CreateSocket (wifiStaNodes.Get (**numNode**), TcpSocketFactory::GetTypeId ());
	tcpSocket->TraceConnectWithoutContext("CongestionWindow", MakeBoundCallback (&CwndChange,**numNode**));
	tcpSocket->TraceConnectWithoutContext ("NextTxSequence", MakeBoundCallback (&NextTxSequence,**numNode**));
	Ptr<MyApp> app = CreateObject<MyApp> ();
	app->Setup (tcpSocket, sinkAddress, 512, 1000000, DataRate ("11Mbps"));
	wifiStaNodes.Get(**numNode**)->AddApplication(app);
	app->SetStartTime (Seconds (nwStart[**numNode**]));
	app->SetStopTime (Seconds (nwStop[**numNode**]));
	stdCBack.str("");
	stdCBack << "/$ns3::NodeListPriv/NodeList/" << **numNode** << "/$ns3::Node/$ns3::Ipv4L3Protocol/SendOutgoing";
	Config::ConnectWithoutContext(stdCBack.str(), MakeBoundCallback(&OutputBytes,**numNode**));
	}
	…

Dica:	Utilize o ConfigStore (item 5) para verificar se está capturando corretamente o código de cada nó.
Problemas:	1) É possível que o número de nós venha a ultrapassar a quantidade de endereços IPs disponíveis, portanto mudar para uma rede que suporte mais hosts, exemplo: 192.168.1.0/255.255.255.0 para 172.16.0.0/255.255.0.0, que de 254 hosts pode suportar até mais que 65000 hosts. 2) É possível que o número de nós também venha a ultrapassar o espaço de mobilidade que foi pré-definido, por tanto, basta alterar: mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel", "Bounds", RectangleValue (Rectangle (**-100, 100, -100, 100**)));

8) A simulação via UDP leva um tempo maior de execução do que uma simulação via TCP?

Explicação:	É normal que isso ocorra, porém é possível reduzir esse tempo de execução observando o tamanho dos pacotes enviados e, também, a taxa de transmissão de dados “DataRate”. Por exemplo, se a rede abaixo é 802.11b, a taxa de **54 Mbps** irá sobrecarregar o nó de origem e causar lentidão na simulação; sendo assim, o certo seria configurar para **11 Mbps**.
Exemplo de sobrecarga:	wifi.SetStandard (WIFI_PHY_STANDARD_80211b); … app->Setup(tcpSocket, sinkAddress, pktSize, 10000000, DataRate ("**54Mbps**")); … (configurar para 11 Mbps ou dimensionar)

9) Como modificar os valores de SIFS e DIFS?

Explicação	Até o momento desta anotação, o valor de DIFS pode ser modificado indiretamente pelo valor de Slot ou pelo valor de PIFS. Já o valor de SIFS pode ser modificado diretamente. Para saber qual é o path de Slot, SIFS e DIFS, basta utilizar o ConfigStore conforme o item 5)

Solução:	uint32_t nwSlot = 20; // Padrão 802.11b 20 Microsegundos
	uint32_t nwSifs = 10; // Padrão 802.11b 10 Microsegundos
	uint32_t nwPifs = nwSifs + nwSlot; // Padrão 802.11b 30 Microsegundos
	…
	Config::Set ("/NodeList/*/DeviceList/*/ $ns3::**WifiNetDevice/Mac/Slot**", TimeValue (MicroSeconds (nwSlot)));
	Config::Set ("/NodeList/*/DeviceList/*/ $ns3::**WifiNetDevice/Mac/Sifs**", TimeValue (MicroSeconds (nwSifs)));
	Config::Set ("/NodeList/*/DeviceList/*/ $ns3::**WifiNetDevice/Mac/Pifs**", TimeValue (MicroSeconds (nwPifs)));

10) Como capturar o endereço de origem dos pacotes?

Explicação:	Em algumas simulações é necessário capturar a origem dos dados, geralmente UDP, e isso pode ser feito através do trace “LocalDeliver”.

Solução:	static void TransmissionBytes (const Ipv4Header &iph, Ptr<Packet const> pkt, uint32_t iface) {
	  NS_LOG_UNCOND ("trby " << **iph.GetSource()** << "\t" << Simulator::Now().GetSeconds() << "\t" << pkt->GetSize());
	}
	…
	stdCBack.str("");
	stdCBack << "/$ns3::NodeListPriv/NodeList/0/$ns3::Node/ $ns3::**Ipv4L3Protocol/LocalDeliver**";
	Config::ConnectWithoutContext(stdCBack.str(), MakeCallback(&TransmissionBytes));

11) Como modificar a “semente” do random a cada simulação?

Explicação:	Se executar um script NS3 por duas ou mais vezes, os resultados serão exatamente os mesmos, devido ao valor de SeedRandom. Para que cada execução do mesmo script tenha valores randomicos diferentes, basta adicionar antes de qualquer configuração do NS3 o seguinte código:
Solução:	int main (int argc, char *argv[]) { ns3::RngSeedManager::SetSeed(**5**);

Essas anotações foram feitas com base na experiência dos alunos (Eduardo, Erich, Hermano, Jamil, José Otávio e Wagner) durante as atividades da disciplina de SADRC/2015 da UTFPR (Prof. Mauro).