Hands-on 6

O sixth é uma versão alterada do fifth onde automatizamos a criação do arquivo pcap e da extração do output em um arquivo de texto.

O primeiro passo para alcançar isso foi adicionar uma linha extra em cada sink para escrever as informações do trace em uma stream de dados no caso do CwndChange ou diretamente em um objeto de arquivo Pcap no caso do RxDrop.

```
static void
CwndChange (Ptr<OutputStreamWrapper> stream, uint32_t oldCwnd, uint32_t newCwnd)
{
  NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << "\t" << newCwnd);
  *stream->GetStream () << Simulator::Now ().GetSeconds () << "\t" << oldCwnd << "\t" << newCwnd << std::endl;
}

static void
RxDrop (Ptr<PcapFileWrapper> file, Ptr<const Packet> p)
{
  NS_LOG_UNCOND ("RxDrop at " << Simulator::Now ().GetSeconds ());
  file->Write (Simulator::Now (), p);
}
```

Com os objetos que representam os arquivos criados, agora temos que criar os arquivos em si e fazer a criação dos objetos que irão representá-los no código com a chamada das funções objetos no construtor para fazer a ligação da saída do programa com os arquivos, para isso se usa os helpers de AscIi (para arquivos de texto) e de Pcap.

```
AsciiTraceHelper asciiTraceHelper;
Ptr<OutputStreamWrapper> stream = asciiTraceHelper.CreateFileStream ("sixth.cwnd");
ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow", MakeBoundCallback (&CwndChange, stream));

PcapHelper pcapHelper;
Ptr<PcapFileWrapper> file = pcapHelper.CreateFile ("sixth.pcap", std::ios::out, PcapHelper::DLT_PPP);
devices.Get (1)->TraceConnectWithoutContext ("PhyRxDrop", MakeBoundCallback (&RxDrop, file));
```