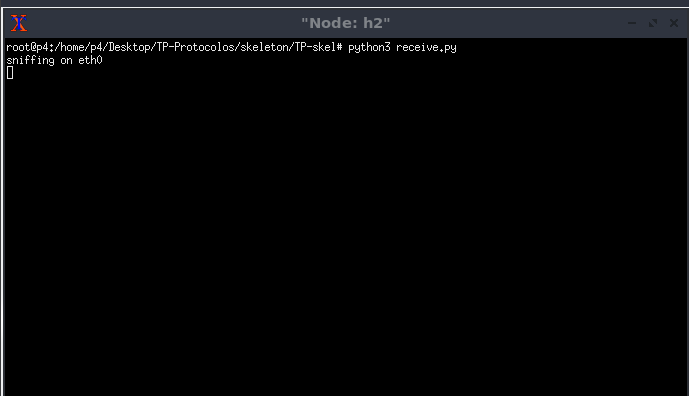
* **Headers :**
  + **ethernet\_t:** Cabeçalho Ethernet padrão com endereços MAC de origem e destino e tipo de protocolo.
  + **ipv4\_t:** Cabeçalho IPv4 padrão contendo informações sobre endereços IP de origem e destino, TTL, checksum, etc.
  + **int\_pai\_t:** Cabeçalho INT para o "pai", que armazena a quantidade de "filhos", o tipo de próximo cabeçalho, e uma flag para identificar se o MTU foi excedido.
  + **int\_filho\_t:** Cabeçalho INT para cada "filho", que coleta dados específicos de cada switch, incluindo portas de entrada e saída, e timestamps de ingress e egress.
* **Parser :**
  + O parser inicia a análise do pacote a partir do cabeçalho Ethernet.
  + Se o pacote for IPv4, o parser extrai o cabeçalho IPv4.
  + Se o pacote for do tipo INT, o parser extrai o cabeçalho int\_pai\_t e, em seguida, analisa uma quantidade de int\_filho\_t de acordo com o campo Quantidade\_Filhos do int\_pai\_t. Quando todos os int\_filho\_t são extraídos, o cabeçalho IPv4 é extraído.
* **Ingress:**
  + ipv4\_forward: Esta ação ajusta os endereços MAC do pacote para a próxima etapa e decrementa o TTL.
  + ipv4\_lpm: Tabela que realiza encaminhamento baseado em endereços IPv4, onde a ação padrão é descartar pacotes.
  + A lógica aplica o encaminhamento IPv4 se o pacote contiver um cabeçalho IPv4 válido.
* **Egress:**
  + **add\_int\_pai:** Se o cabeçalho int\_pai\_t não estiver presente, ele é criado. Este cabeçalho inicializa o número de "filhos" para 0 e substitui o tipo de EtherType para indicar um pacote INT.
  + **add\_int\_filho:** Para cada switch, um novo int\_filho\_t é adicionado com informações específicas, como IDs de switch, portas de entrada e saída e timestamps de entrada e saída.
  + Se o comprimento do pacote somado ao tamanho de um int\_filho\_t for menor que o MTU, um int\_filho\_t é adicionado. Caso contrário, o campo MTU\_Overflow no int\_pai\_t é marcado para indicar que o pacote excede o MTU.
* **Checksum:**
  + **MyComputeChecksum:** Computa o checksum do cabeçalho IPv4 antes de o pacote ser emitido.
* **Deparser:**
  + O deparser emite os cabeçalhos na ordem correta: Ethernet, int\_pai\_t, int\_filho\_t, e IPv4.3
* **Switch:**
  + Finalmente, o programa principal define o switch, integrando todas as fases: parser, checksum, ingress, egress, checksum final, e deparser.

Funcionamento Geral:

Este switch coleta dados sobre cada salto que um pacote dá na rede. Isso é feito encapsulando os dados no cabeçalho int\_filho\_t, que é anexado ao pacote original. Esses dados podem incluir informações como a porta pela qual o pacote entrou e saiu, e timestamps para medir latência. O cabeçalho INT (int\_pai\_t) é usado para controlar quantos desses blocos de informação foram adicionados e para monitorar se o MTU do pacote foi excedido.

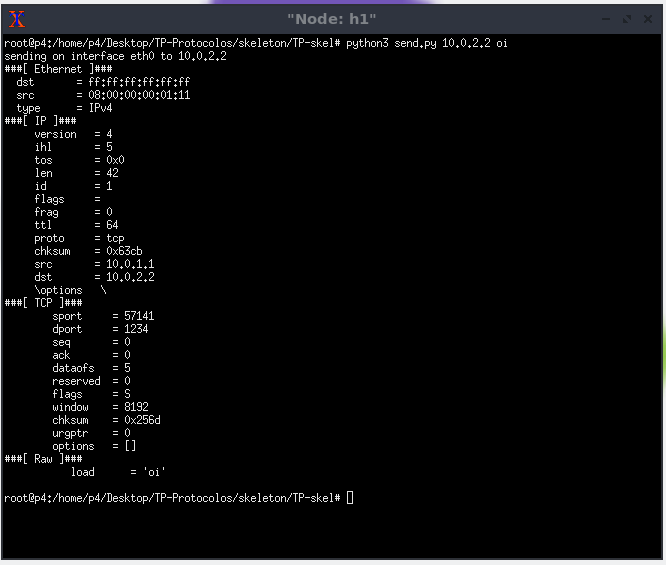
Resultado:



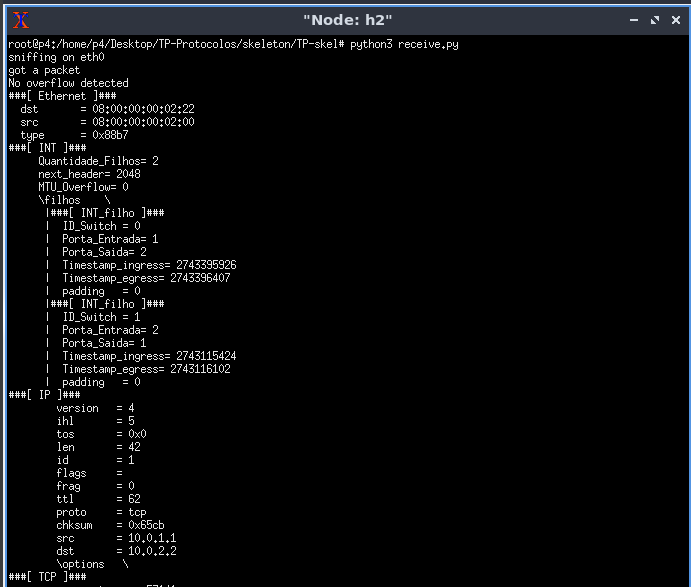
h2 antes de receber o envio de h1

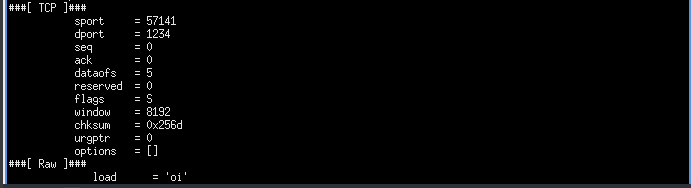


A mensagem que sera enviada para h2

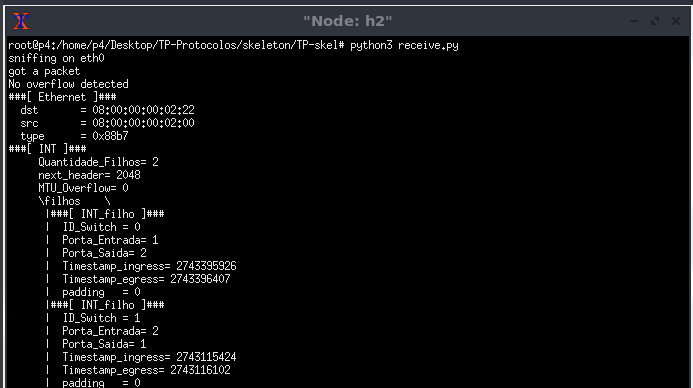


A mensagem enviada por h1 para h2

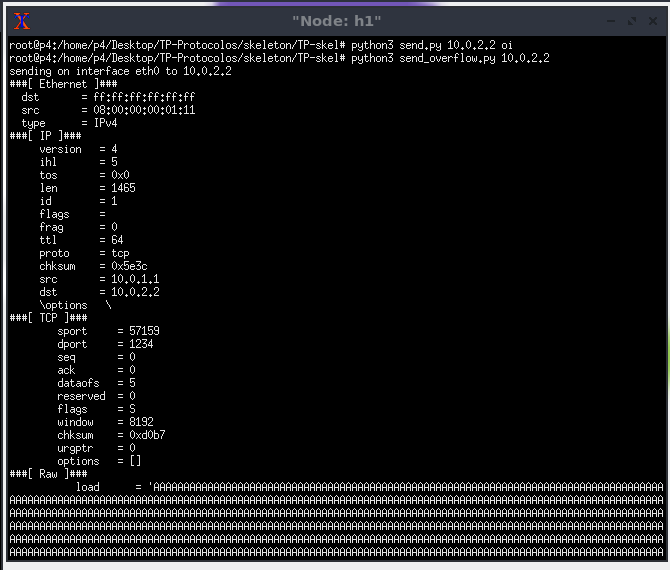




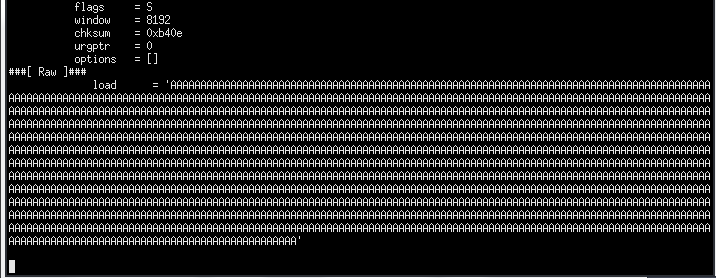
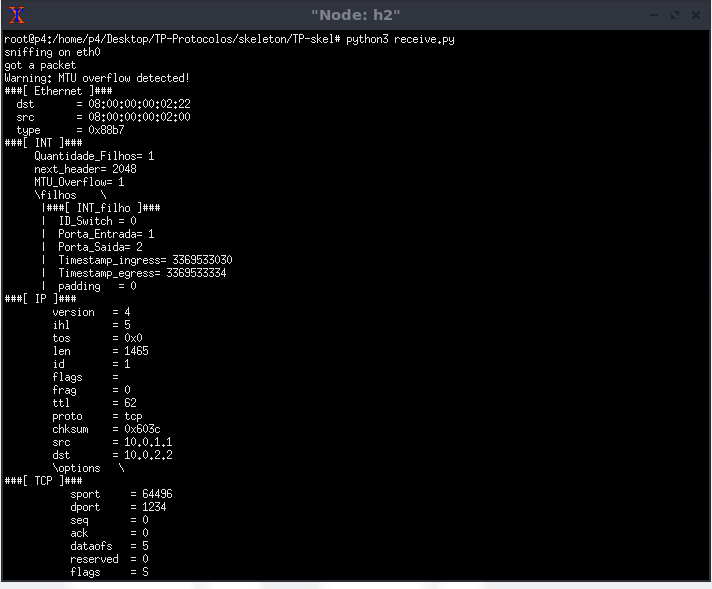
A mensagem recebida por h2 um header int pai e dois int filhos adicionados a mensagem recebida que não estavam no pacote enviado



Na linha apos got a packet o sinal de que não houve um overflow do MTU dois filhos adicionados



Envio de uma novam apartir de h1 d mensageesta vez uma mensagem grande o suficiente para para haver um overflow do MTU



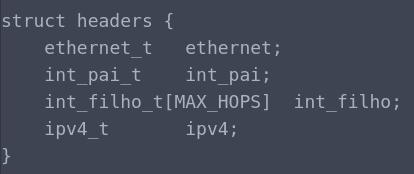
Na linha após got a packet aviso de overflow do MTU detectado desta fez apenas o int pai e um filho foram adicionados diferente da vez anterior.

Problemas durante o desenvolvimento do trabalho:

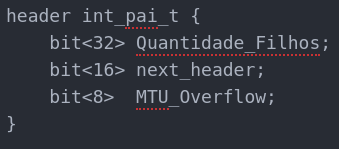
A principal dificuldade foi ter um código inicial que funcionasse por causa da falta de experiência com a linguagem p4 tive uma dificuldade inicial de entender o que precisava ser feito inicialmente, mas após algumas tentativas consegui adicionar um cabeçalho int pai fazer ele ser reconhecido pelo receiver após isso adicionar os filhos e oe controle do MTU foram bem mais simples. Outro problema foi o ID do switch não achei uma forma de acessá los então uso o número de filhos do int pai no momento em que chega no switch para identificar os switch pelos quais o pacote passou.

Passos para a implementação dos requisitos:

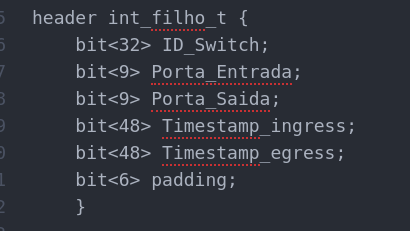
Coloquei os headers int pai e int filho entre as camadas ethernet e ip porque achei que seria mais simples não mexer no pacote ip por causo do checksum



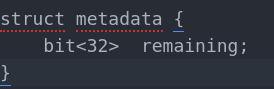
O header int pat possui o numero de filhos usado pelo parser para processar corretamente todos os filhos encadeados next header aponta para o pacote ipv4 original da mensagem MTU overflow é a flag de overflow



As informação que o int filho coleta

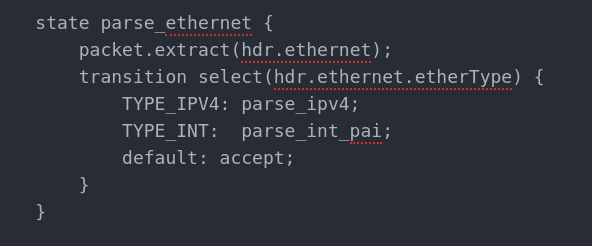


Remaining é utiliza no parser para processar os int filhos

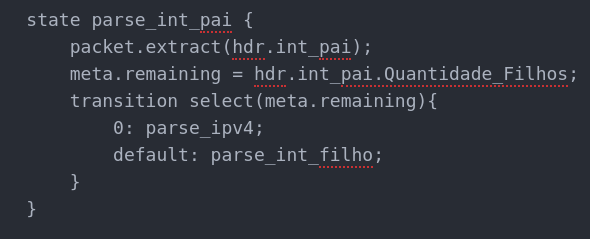


Mudanças no parser :

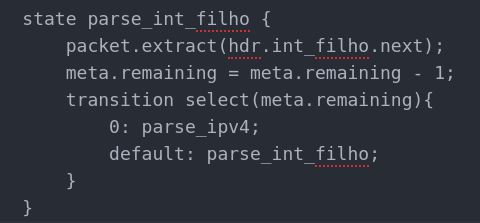
parse\_ethernet agora precisa ter uma transição para o caso o próximo header seja um int pai o pacote int pai é identificado por possuir tipo igual a 0x88B7



Parse int pai coloca a quantidade de filhos na struct metadata para não alterar os campos originas do pacote



Processa cada filho ate remaining chegar a 0 após isso continua para processar o pacote ipv4



Egress adiciona o header int pai se não houve um e caso não cause um estouro no MTU adiciona um int filho no caso de estouro set a flag MTU\_Overflow para 1

