UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Disciplina INF01002 - Protocolos de Comunicação

Alunos: **André Dexheimer Carneiro** **Matrícula:** **00243653**
Rubens Ideron dos Santos **00243658**

Trabalho Final Parte 2 - Documentação

1. Lógica de Funcionamento do Mecanismo

Tomamos como base o trabalho entregue na parte um. Em cima dele, implementamos a lógica de separação dos cabeçalhos inseridos com dados de telemetria e o payload do pacote. Os dados de telemetria são enviados sem o conteúdo da mensagem para um *host* fixo, no nosso caso h1, enquanto o payload segue seu caminho original até o host destino sem os cabeçalhos int.

Ao atingir o *last hop,* isto é, o último roteador antes de atingir o destinatário final, o pacote é clonado usando a primitiva *clone* do *V1Model* ao final da etapa *Ingress.*

O pacote original é usado para enviar as informações de telemetria e, portanto, nele são feitas as seguintes alterações:

- O endereço IP de destino é alterado para o host H1
- O endereço MAC de destino é alterado para o atual remetente
- O campo *evil bit* do IPv4 é decrementado em 4 para assumir o valor original antes da inserção dos cabeçalhos de telemetria.
- O campo *protocol* do IPv4 é alterado para 145 para sinalizar o pacote como um pacote de telemetria, assim evitamos que novos cabeçalhos de telemetria sejam adicionados nele.

O pacote clonado está prestes a ser entregue ao seu destinatário, portanto basta remover os cabeçalhos de telemetria, decrementar o TTL e ajustar o *evil bit* da mesma maneira como foi feito para o pacote original.

2. Experimentos

A validação do funcionamento é simples, basta enviar um pacote TCP entre quaisquer dois *hosts* e observar que o *payload* do tcp chega apenas ao destinatário original da mensagem, enquanto o host H1 recebe apenas os cabeçalhos de telemetria.

O método para a validação dos cabeçalhos de telemetria é a mesma descrita na parte um do trabalho.

3. Problemas encontrados durante a implementação

- Como clonar o pacote

Pesquisando na documentação na internet e com a ajuda do Lucas encontramos duas maneiras: alguma das primitivas *clone* ou usando *multicast*. Optamos pelo clone por ter mais documentação disponível.

- Configuração dos Switches para encaminhar os clones a determinada porta
  Isto foi difícil de encontrar na documentação. Para que as primitivas clone e clone3 funcionem apropriadamente os switches devem ser configurados pelo P4Runtime. Para isto que servem os arquivos sn-commands.txt e config.sh. Descobrir como utilizá-los apropriadamente não foi fácil, visto que não encontramos nenhum tutorial ou documentação a respeito. Os comandos P4Runtime devem ser colocados nos arquivos commands.txt e, para que façam efeito, o config.sh deve ser rodado no terminal de algum switch aberto com xterm.

- Como diferenciar pacotes de telemetria de pacotes de dados

Utilizamos o valor 145 no campo *protocol* do cabeçalho IPv4 pois ele não é utilizado oficialmente até hoje.
(https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml)

- Pacotes adicionais sendo recebidos pelos hosts

Esta foi a etapa de depuração mais difícil do trabalho e teve a solução composta por alguns itens que faltavam na implementação. São eles:

- O campo MAC destino não estava sendo corretamente definido no pacote que continha os dados de telemetria.
- A primitiva *clone* pode ser usada nas configurações *egress to egress* ou *ingress to egress*. O que altera o ponto no processamento em que temos que ajustar os conteúdos dos pacotes.
- Pacotes ICMP que estavam trafegando na rede. Por muito tempo, pensamos que fossem pacotes clonados que seguiam um caminho incerto, mas depois descobrimos que eram pacotes ICMP. Como não eram relevantes ao funcionamento do mecanismo, tinham de ser

ignorados no código P4 para que não fossem tratados como dados ou telemetria.

- Descobrimos que pacotes ICMP trafegavam pela rede através de um programa Python de depuração. Se trata de uma modificação do receive.py fornecido pelo professor. Esta modificação monitora todas as portas dos switches e deve ser rodada no terminal de qualquer switch. Através dela descobrimos que o campo protocolo do cabeçalho IPv4 era 1, correspondente ao protocolo ICMP (https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml).

- Documentação e ferramentas de depuração

A documentação sobre P4, embora extensa, não cobre alguns pontos chave para a compreensão do mecanismo da primitiva *clone.* Além disso, não existe uma ferramenta acessível para a depuração do código P4.

Além disso, certas primitivas não funcionam no ambiente fornecido devido à versão da linguagem P4 utilizada (14 vs. 16). É o caso da primitiva clone_ingress_pkt_to_egress por exemplo.

4. Passos da Implementação dos Requisitos Funcionais

Usamos a porta de saída do *switch* para identificar se o pacote seria encaminhado a um *end-host* ou não (também preparamos o código para utilizar um campo da tabela de encaminhamento para esta finalidade, de forma que funcione com topologias de rede genéricas). Desta forma, ao final do *ingress processing*, foi usada a primitiva *clone* no modo *Ingress to Egress*, que copia o pacote como era logo ao chegar no switch e o encaminha diretamente ao egress processing, sem passar pelas modificações do ingress processing. Os cabeçalhos INT e o payload do TCP são removidos usando chamadas ao método *setInvalid* após diferenciar entre o pacote de dados e o pacote de telemetria.

A diferenciação entre o pacote clonado e o original é feita usando o valor de *standard_metadata.instance_type,* que recebe o valor 1 para o pacote clonado e 0 para o original. Uma vez que o pacote de telemetria segue pela rede até o seu destinatário, ele é diferenciado de pacotes de dados pelo valor do campo *protocol* no cabeçalho IP, que é alterado para 145 quando o pacote é duplicado.

Para receber os pacotes de telemetria, usamos o mesmo *script receive.py*, que exibe todo o novo pacote e seu conteúdo.