Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN

A introdução das SDNs no mundo das redes, gerou um novo paradigma em relação à maneira como as redes são programadas. Tendo em conta que o OpenFlow possibilitou a divisão dos switches em dois segmentos, *control plane* e *data plane*, permitiu que houvesse uma maior investigação nesta área. Porém, estas investigações, maioritariamente, destinam-se apenas à programação de controladores (*control plane*) o que levou a estes investigadores a procurarem melhorias para o segundo segmento.
Este artigo foi proposto com o intuito de ultrapassar duas limitações relativamente aos *switching chips* e protocolo de OpenFlow. A primeira limitação refere-se há pouca flexibilidade do *hardware* dos *switches*, em que o processo de *Match-Action* é permitido em apenas alguns campos fixos. A segunda diz que as especificações do OF apenas definem um certo reportório de ações de processamento de pacotes.

Com base nestes problemas, os autores propuseram a arquitetura RMT (*Reconfigurable Match Tables*), em que foram identificados o conjunto mínimo de ações primitivas para especificar como é que os cabeçalhos são especificados em *hardware*. Este modelo permite que o plano de reencaminhamento possa ser modificado em "campo" sem ser preciso alterar o *hardware*. Permite modificar todos os campos do cabeçalho mais compreensivamente do que no OF.

Eles pretendiam implementar o "mecanismo" de *Match-Action* no *hardware* de modo a tirar proveito do paralelismo e do *pipelining*, enquanto existem restrições relativas à memória dos chips.

A arquitetura RMT precisa de um *parser* reconfigurável para poder repartir o *header* num vetor de campos, para que, através de uma sequência de fases lógicas, se possa efetuar a correspondência nas tabelas (*pipelining*) - para cada campo, existe uma fase. Este processo, pode ser realizado em *multi-thread* (paralelismo), ou seja, pode existir vários vetores de *headers* a serem correspondidos. No final deste processo, o vetor é recombinado e readicionado ao pacote, sendo reencaminhado ao canal respetivo. Durante este processo, poderá haver modificações nos campos, as quais são feitas através de uma instrução geral (VLIW) que só opera aos campos correspondentes. Isto é, cada campo poderá ter uma operação diferente na mesma instrução.

Relativamente à implementação, os autores defendiam que a arquitetura devia consistir num grande número de estados de pipelines físicos, que num número menor de estados lógicos de RMT, que podem ser mapeados, isto dependendo das necessidades dos recursos de cada estado lógico. Esta implementação foi motivada por três componentes: *Factoring State,* diz que o reencaminhamento de um *router* tem vários estados e que cada um desses estados usa uma tabela separada, mas se combinarmos isto apenas numa tabela, o resultado é o cruzamento de estados e como estes são processados sequencialmente (com dependências), usar um pipeline físico é o recurso mais lógico; *Flexible Resource Allocation Minimizing Resource Waste* resume-se à flexibilidade em alocar estados físicos em estados lógicos (os recursos necessários para um estado lógico pode variar significativamente); *Layout Optimality* refere que a um estado lógico pode ser designado mais memória se este tiver múltiplos estados físicos adjacentes.

As vantagens do RMT é que este permite que o plano de dados possa ser reconfigurado de certos aspetos. Muitos protocolos foram propostos (e.g PBB, VxLAN, STT, etc), contudo sem uma arquitetura como o RMT, seria necessário criar um novo *hardware* que se adequa-se a estes protocolos.

Um dos grandes problemas do modelo RMT é que pode não ser implementável para grandes velocidades.