Data Center TCP (DCTCP)

A *Cloud* tem sido uma tecnologia bastante requerida nos últimos tempos, logo tem existido uma necessidade de promover e melhorar as características da mesma, tais como, a disponibilidade e alto desempenho a pouco custo. Esta, geralmente é constituída por *data centers* (conjuntos de clusters) em que a sua comunicação é feita através do protocolo TCP, que requerem certas propriedades: pouca latência para pequenos fluxos de dados, alta tolerância a picos de fluxos e uma grande utilização para grandes fluxos.

Existem duas contribuições descritas pelos autores. A primeira incide na medição e avaliação do tráfego num *data center*, a fim de entender como é que o TCP se comporta, bem como as necessidades e os padrões das aplicações, como por exemplo a baixa latência. Na segunda sugerem o protocolo DCTCP, que é uma variação do TCP, que tem como objetivo melhorar algumas desvantagens que o TCP apresenta. Com base na primeira contribuição também se pretende ter meios de comparação com o novo protocolo.

Com base nas medições feitas, chegaram à conclusão que o TCP não preenchia os requisitos necessários para as propriedades de um *data center* verificando falhas na utilização deste.
Existem três casos de falha no uso do TCP: o *incast*, o *queue buildup* e o *buffer pressure*. Basicamente, estas falhas resultam num *overflow* e consequentemente, na perda de pacotes nos *switches*.
No caso do *incast*, se um *switch* receber vários fluxos de pacotes simultaneamente num curto período de tempo, pode exceder a memória deste resultando na perda de pacotes. Isto pode ocorrer tanto com grandes como com pequenos fluxos. Foram propostas soluções para resolver este problema tal como, diminuir o tamanho máximo dos pacotes no recetor de forma a não exceder a memória; outra solução considerada, e que o DCTCP implementa, é o acréscimo do *jitter* para a dessincronização das respostas, isto faz com que os *timeouts* sejam evitados pois reduz o tempo de resposta quando existe grande latência, porém isto tem um custo de aumentar o tempo médio de resposta.
No segundo caso, existe a situação de fluxos pequenos e fluxos grandes usarem a mesma *queue,* o que pode provocar latência nos fluxos pequenos (estando estes atrás dos pacotes de maior fluxo) mesmo que não haja perda de pacotes. Como esta latência é causada pela construção das filas de espera, a solução é reduzir o tamanho destas.
Por fim, existe o problema de grandes fluxos ocorrerem em determinadas portas e ter impacto nas portas em que o fluxo é menor. Isto resulta numa perda de pacotes e *timeouts*, como no *incast* mas sem requerer fluxos sincronizados.

Os autores resolveram estes problemas com o desenvolvimento de um algoritmo do DCTCP, que tem como objetivo obter uma alta tolerância a picos de pacotes, reduzir a latência e ter um elevado rendimento. Este algoritmo fornece três componentes. A primeira componente marca os pacotes no recetor quando o *buffer* chega a um determinado limite (através da notificação ECN - *Experience Congestion Notification*). Na segunda componente, o recetor define a flag ACK numa série de pacotes, que reenviará ao emissor, isto até receber a confirmação de que o emissor tenha recebido a notificação da congestão. No lado do emissor, este recebe os pacotes marcados pelo recetor, envia um ACK da receção do ECN e reduz o número de pacotes que envia.

Com base nos resultados obtidos, os autores concluíram três pontos-chave: o primeiro foi que se o *data center* deles usar DCTCP poderia aguentar dez vezes mais pedidos de resposta e dez vezes mais fluxos em *background* com melhor desempenho que o do TCP; a segunda conclusão foi que usar *deep buffered switches* melhora o desempenho do tráfego, isto faz com que o desempenho de pequenas transferências piore devido ao *queue buidup*. Para terminar, embora o RED melhore o desempenho de pequenas transferências, este não melhora o desempenho do tráfego devido à variação do tamanho da fila.

Antes do desenho do algoritmo DCTCP os autores avaliaram esquemas AQM (*Active Queue Management*), como o RED ou o PI que não modificam os mecanismos de controlo de congestão do TCP. Eles descobriram que estes não funcionavam bem quando a multiplexagem estatística é baixa e quando existe muito tráfego por um curto período de tempo, o que acontece em *data centers*. Eles avaliaram o

desempenho do RED, bem como do PI relativamente ao DCTCP e puderam concluir, em ambas as análises que o DCTCP tem uma taxa de transferência alta e poucos atrasos num ambiente com baixa multiplexagem estatística.