[2페이지]

Improving network availability with Protective ReRoute 논문을 요약하면 기존에 사용되던 네트워크보다 가용성을 더 높이기 위해 Protective ReRoute(PRR)을 사용한다가 되겠습니다. 이 논문에서 네트워크 가용성을 이러한 수식으로 나타냈는데, MTBF는 실패와 실패 사이의 평균 시간, MTTR은 복구되는 평균 시간을 나타냅니다. 실패 사이의 평균 시간이 길다는 것은 그만큼 실패가 자주 일어나지 않는다는 것을 의미하게 되고 복구 시간이 짧다면 그만큼 사용자가 겪는 Outage가 줄어들게 되어 가용성이 늘어나게 됩니다.

[3페이지]

논문의 핵심인 Protective ReRoute에 대해 알아보겠습니다. 간단한 예시를 들자면 자동차는 기존에 중앙의 길로 가려했지만 outage 즉, 중단 혹은 black hole 처럼 경로로 진행할 수 없어 경로를 재설정해야 합니다. 결국 자동차는 주황색의 경로로 경로를 재설정해 원하는 목적지에 도착할 수 있게됩니다.

[4페이지]

PRR에 대해서 더 자세히 알아보겠습니다. 현대 네트워크는parallel 즉, 병렬적 링크를 추가하여 용량을 늘리는 방식을 채택합니다. 이를 가능하게 하기 위해 ECMP가 경로에 전송 flow를 분배하게 됩니다. 여기서 ECMP는 Equal Cost Multipath로 다수의 경로가 목적지, 거리, 우선순위 값이 동일할 경우, 경로가 모두 active되어 트래픽을 분배하게 되는 기술입니다. 아래 그림에는 출발지부터 도착지까지 가는 여러 경로가 존재합니다. 데이터는 이러한 여러가지 경로를 타고 원하는 지점에 도달할 수 있게 됩니다.

[5페이지]

Client가 데이터를 전송하기 위해 FlowLabel이 1인 경로를 선택한 경우를 살펴보겠습니다. 여기서 FlowLabel은 IPv6 header에 안에 존재하며, 20bit로 이루어져 있습니다. FlowLable은 일종의 식별자로서 특정 트래픽 경로를 식별하는 데 사용됩니다. FlowLabel이 1인 경로는 Site B에서 Black hole 즉, 어떠한 이유로 전송에 실패를 하게 됩니다. 이는 네트워크 연결이 끊어져 사용자 입장에선 outage로 다가오게 됩니다. outage가 발생해도 host는 다른 정상 작동하는 경로를 가지고 있기 때문에 단순히 그러한 경로들로 바꿔 연결을 복구할 수 있습니다.

[6페이지]

사용자의 outage를 줄이기 위해 빠른 경로 재설정이 필요합니다. PRR은 실패가 감지되면 경로 재설정을 하게 됩니다. 기존 ECMP에선 host가 경로를 변경할 수 없어 IP 주소나 port를 변경해야 했지만 PRR에서 경로 재설정은 단순히 host가 ECMP안의 IPv6의 header인 FlowLabel을 변경하는 것만으로도 가능합니다. 이 예제에서는 black hole이 발생한 기존 FlowLabel 1에서 FlowLabel 2로 변경하며 원하는 위치까지 도달한 모습을 확인할 수 있습니다. 만약 한번에 복구가 안된다면 다른 정상 작동하는 경로로 FlowLabel을 바꿔 복구를 다시 시도하게 됩니다.

[7페이지]

그렇다면 실패에는 어떤 유형이 있고 이를 어떻게 복구할 수 있는지 알아보겠습니다. 실패에는 단방향 실패와 양방향 실패 2가지가 존재합니다. 첫번째 경우인 단방향 실패는 다시 forward, reverse 두 개의 경우로 나누어집니다. Forward는 Data를 전송할 때 문제가 생기는 것으로 RTO가 발생되게 됩니다. Data를 전송할 때의 실패는 RTO가 발생 하냐 안하냐로 판단하게 되고, RTO가 발생되면 경로를 재설정합니다. 예제에선 첫 전송이 실패해 경로를 바꿨지만 여전히 실패하여 한번 더 경로를 바꾼 결과 복구가 되어 ACK까지 잘 도착한 모습입니다. 두번쨰 경우인 역방향 실패는 중복을 기준으로 실패를 판단하게 되는데, 단방향 실패보다 더 많은 overhead가 요구됩니다. 예제에선 Data는 전송을 성공하였지만 ACK 전송에서 문제가 생깁니다. 하지만 전송자는 Data에서 실패가 되었는지, ACK이 실패했는지 알 수 없기 때문에 RTO가 발생하게 되고 Data의 경로가 재설정되게 됩니다. Receiver는 똑같은 데이터를 다시 받게 되고 중복이라는 것을 감지합니다. 중복을 통해 reverse 즉, ACK의 경로를 재설정하게 됩니다. 또 다시 RTO가 발생하여 Data 경로가 재설정되고, ACK까지 잘 받게 되며 복구가 완료됩니다. 양방향 실패는 허위 RTO와 harmful repathing으로 단방향 실패보다 더 많은 overhead가 사용됩니다.

[8페이지]

한 상황 예제를 통해 repathing이 어떻게 이루어 지는지 확인해보겠습니다. 첫 전송과 두번째 전송에서 data 전송이 실패가 일어나 경로가 재설정 됩니다. 경로가 재설정 되면서 data는 잘 전송되게 되고, 이번엔 ACK 전송에서 문제가 생기지만 중복이 감지되지 않으므로 Ack 경로를 변경하지 않고 RTO가 다시 발생합니다. 여기서 ACK이 경로 재설정이 필요한데, RTO가 발생하면서 data가 오히려 다시 경로를 설정하지만 심지어 전송에 실패하게 되는데 이를 harmful repathing이라 하고, 패킷 손실과 서비스 품질의 저하를 일으키게 됩니다. 이후 receiver에서 중복을 감지하고 ACK의 경로를 재설정하게 되고 마지막에 경로에서 경로가 모두 복구된 것을 확인할 수 있습니다.

[9페이지]

시뮬레이션 결과를 통해 실패의 종류에 따른 outage의 비율을 확인해보겠습니다. 첫번째 그래프에선 UNI는 단방향, BI는 양방향 실패로, 50% outage인 단방향 실패와 25%,25% 양방향 실패가 거의 비슷한 실패 비율을 나타내고 있습니다. 단방향 실패가 복구될 확률은 1/2, 양방향 실패가 복구될 확률은 9/16으로 거의 비슷한 것 그래프에서도 알 수 있습니다. 오른쪽 그림에서도 Forward, Reverse, 양방향 순으로 점점 실패 비율이 올라가고 있음을 확인할 수 있습니다. 또한 두 그래프의 꼬리가 점점 내려가는 것을 통해 시간이 경과함에 따라 복구가 촉진되고 있음을 알 수 있습니다.

[10페이지]

이번엔 PRR에서 RTO가 실패 비율에 미치는 영향을 알아보겠습니다. 50%가 outage가 발생한 아래 그래프에선 RTO가 짧으면 짧을 수록 실패 비율이 줄어드는 것을 확인할 수 있습니다. RTO가 짧을 수록 빠른 실패 탐지와 복구가 가능해지기 때문에 실패 비율이 줄어들게 됩니다. 여기서 0초일때 실패 비율이 0.3을 넘지 않는 것을 확인할 수 있는데, 이는 timeout전에 경로가 이미 복구되었음을 암시합니다. Fault duration은 평균적으로 지속되는 실패 시간이라고 합니다.

[11페이지]

앞서 살펴본 바에 따르면 PRR은 경로를 재설정 할 때 다른 네트워크 구성요소에 의존하지 않고 FlowLable을 사용하여 local적으로 repathing을 진행합니다. PRR은 패킷 안의 FlowLabel 값을 무작위로 변경하여 경로를 재설정하게 됩니다. ECMP에서 다양한 입력이 각 스위치에서 사용 가능한 다음 홉을 무작위로 고르는 것과 동일합니다. 그림은 25% 확률로 실패가 발생하는 경우를 설명합니다. 이용 가능한 4개의 경로 중 하나를 선택했을 때 성공할 확률은 75%가 됩니다. 만약 p 확률로 실패가 발생하고 N번의 repathing이 발생하면 p^N의 확률로 repathing이 성공하게 됩니다. Outage로 인해 실패한 경로의 비율은 두 쌍이 어떻게 구성되어 있는 지와 시간의 경과가 지남에 따라 계속하여 바뀌게 됩니다.

[12페이지]

하지만 repathing은 cascade 문제를 반드시 피해야합니다. Cascade란 한 문제가 전체 시스템에 다른 문제를 연속적으로 일으키는 것으로, PRR은 cascade 관련 문제에서 fast routing보다 더 점진적이고 원활하게 repathing을 진행하게 됩니다. 그림에선 여러 군데에서 경로를 재설정하기 위해 무작위로 FlowLabel을 골랐지만 어느 경우에 한 곳에 트래픽이 몰려 결국에는 또 다른 문제 black hole이 발생한 cascade 경우입니다. 이는 결국 다시 repathing이 필요하게 됩니다.

[13페이지]

PRR에서는 이를 해결하기 위해 각각의 TCP 연결이 outage에 대해 독립적으로 대처하게 됩니다. 이를 통해 반응 시간이 RTO 시간 단위로 분산되게 됩니다. TCP 연결은 RTO 시간 동안 대기 상태에 돌입하여 RTO 시간 내에 이를 감지하고 다시 전송을 시도하여 불안정한 네트워크에서도 점진적으로 실패를 감지하고 복구할 수 있도록 합니다.

[14페이지]

기존의 방식은 50%의 중단이 발생한다고 가정하면 4개중 2개가 실패, 경로가 재설정 된 2개중 1개가 실패 이렇게 반복되며 결국 약 2배 정도의 overhead가 요구됩니다. 하지만 PRR에서는 이를 해결하기 위해 무작위로 FlowLabel을 선택하긴 하지만 routing 가중치를 사용하여 경로 재지정으로 인해 각 작업 경로에서 예상되는 load의 증가는 outage 비율로 제한되게 됩니다.

[15페이지]

이젠 PRR을 적용한 네트워크와 기존의 네트워크를 비교하면서 얼마나 network 가용성을 향상시켰을지 확인해보겠습니다. 논문에서는 4가지의 case study가 있는데 모두 비슷한 결과를 나타내기 때문에 처음 2개의 결과만 살펴보겠습니다. 이 그래프에선 L3가 네트워크 layer, L7이 애플리케이션 layer을 나타내고, L7/PRR은 PRR이 적용된 L7을 나타냅니다. 기존 L3, L7과 비교하여 PRR이 적용된 네트워크는 inter, intra 두 경우 모두에서 평균 loss 비율이 매우 감소한 것을 확인 할 수 있습니다. L7과 비교하면 거의 100배 정도 복구가 빨라지게 되어 대부분의 소비자는 중단가 발생해도 알아차리지 못하게 됩니다.

[16페이지]

두번째 case는 광 고장으로 physical layer쪽의 문제로 광확적 구성 요소에서 발생하는 실패를 복구하는 PRR의 결과를 나타냅니다. 첫번째 case와 비슷한 그래프 모양을 보이지만 전 case보다 높은 loss 비율이 확인됩니다. Inter L7에서는 약 5배 정도의 loss 비율이 감소한 것을 확인할 수 있고, intra L7에서는 20배 정도 loss 비율이 감소했습니다.

[17페이지]

이젠 PRR이 전체 네트워크 가용성을 얼마나 높였는지 6개월 기간 동안 Google 네트워크의 모든 중단에 대한 측정값을 집계하게 됩니다. 그래프를 확인해보면 PRR이 적용된 막대에서 더 큰 중단 시간 감소가 일어났음을 알 수 있습니다. 중단 시간 감소 즉, MTTR이 줄어들었다는 것이고 더 빨리 경로가 복구되어 더 많은 경로를 사용할 수 있고, 이를 통해 실패가 일어날 확률을 줄여 MTBF가 증가하게 됩니다. 이를 수식에 대입하여 생각하면 처음 제시한 목표인 가용성 향상을 이루게 되었습니다.

[18페이지]

PRR을 정리해보면, ECMP를 위해 IPv6의 FlowLabel을 사용하고, 다양한 경로를 라우팅에 사용하며, host가 FlowLable을 바꾸는 것으로 경로를 재설정할 수 있게 됩니다.