컴퓨터 네트워크의 지연과 처리량 (한국어 번역)

1.4.2 대기열 지연과 패킷 손실

La/R <= 1인 경우를 살펴보겠습니다. 이제 La/R 의 경우를 살펴보겠습니다.

1. 여기서 도착 트래픽의 특성은 대기열 지연에 영향을 미칩니다. 예를 들어, 패킷이 주기적으로 도착하는 경우, 즉 L/R 초마다 하나의 패킷이 도착한다면, 모든 패킷은 대기열이 비어 있는 상태에서 도착하므로 대기열 지연이 없습니다. 반면, 패킷이 주기적으로 버스트 형태로 도착한다면 평균 대기열 지연이 상당히 클 수 있습니다. 예를 들어, N 개의 패킷이 (L/R) * N 초마다 동시에 도착한다고 가정해 봅시다. 그러면:
    - 첫 번째로 전송되는 패킷은 대기열 지연이 없습니다.
    - 두 번째로 전송되는 패킷은 L/R 초의 대기열 지연을 겪습니다.
    - 더 일반적으로, n 번째로 전송되는 패킷은 (n-1) * (L/R) 초의 대기열 지연을 겪습니다.
2. 이 예시에서 평균 대기열 지연을 계산하는 것은 독자들에게 연습 문제로 남겨두겠습니다.

위의 두 가지 주기적 도착 예시는 다소 학문적인 경우입니다. 일반적으로 대기열로의 도착 프로세스는 무작위적입니다. 즉, 도착은 특정 패턴을 따르지 않으며, 패킷들은 무작위 시간 간격으로 분산되어 도착합니다. 이러한 더 현실적인 경우, La/R (트래픽 강도)만으로는 대기열 지연 통계를 완전히 특성화하기에 충분하지 않습니다. 그럼에도 불구하고, 이는 대기열 지연의 정도를 직관적으로 이해하는 데 유용합니다. 특히:

- 트래픽 강도가 0에 가까우면 패킷 도착이 드물고, 도착하는 패킷이 대기열에서 다른 패킷을 만날 가능성이 낮아 평균 대기열 지연은 거의 0에 가깝습니다.
- 반면, 트래픽 강도가 1에 가까우면, 패킷 도착률이 전송 용량을 초과하는 시간 구간이 발생하여 대기열이 형성됩니다. 도착률이 전송 용량보다 낮을 때는 대기열 길이가 줄어들지만, 트래픽 강도가 1에 가까워질수록 평균 대기열 길이는 점점 더 커집니다.

평균 대기열 지연과 트래픽 강도의 질적 의존성은 그림 1.18에 나타나 있습니다. 그림 1.18의 중요한 측면 중 하나는 트래픽 강도가 1에 가까워질수록 평균 대기열 지연이 급격히 증가한다는 점입니다. 강도가 약간만 증가해도 지연은 훨씬 더 큰 비율로 증가합니다. 이는 고속도로에서 경험할 수 있는 현상과 유사합니다. 만약 당신이 자주 혼잡한 도로를 운전한다면, 그 도로의 트래픽 강도가 1에 가까운 상태라는 의미입니다. 이 경우, 약간의 추가 트래픽(예: 사고나 이벤트)만으로도 지연이 엄청나게 커질 수 있습니다.

대기열 지연을 더 깊이 이해하기 위해, 교재 웹사이트에서 제공하는 대기열에 대한 인터랙티브 애니메이션을 방문하는 것을 권장합니다. 패킷 도착률을 높게 설정하여 트래픽

강도가 1을 초과하도록 하면, 대기열이 시간이 지나면서 점차 쌓이는 모습을 볼 수 있습니다.

패킷 손실

지금까지의 논의에서는 대기열이 무한한 패킷을 수용할 수 있다고 가정했습니다. 그러나 실제로는 링크 앞의 대기열은 유한한 용량을 가지며, 이 용량은 라우터 설계와 비용에 따라 크게 달라집니다. 대기열 용량이 유한하기 때문에, 트래픽 강도가 1에 가까워질수록 패킷 지연이 무한대로 증가하지 않습니다. 대신, 패킷이 arrival했을 때 대기열이 가득 차 있으면 라우터는 해당 패킷을 버립니다. 즉, 패킷이 손실됩니다. 이러한 대기열 오버플로는 트래픽 강도가 1을 초과할 때 인터랙티브 애니메이션에서도 확인할 수 있습니다.

종단 시스템의 관점에서 보면, 패킷 손실은 네트워크 코어로 전송된 패킷이 목적지에서 나타나지 않는 것처럼 보입니다. 트래픽 강도가 증가할수록 손실된 패킷의 비율도 증가합니다. 따라서 노드의 성능은 지연뿐만 아니라 패킷 손실 확률로도 측정됩니다. 이후 장에서 논의하겠지만, 손실된 패킷은 종단 간 재전송을 통해 복구될 수 있으며, 이는 모든 데이터가 최종적으로 소스에서 목적지로 전송되도록 보장합니다.

1.4.3 종단 간 지연

지금까지 우리는 단일 라우터에서의 지연, 즉 노드 지연에 초점을 맞췄습니다. 이제 소스에서 목적지까지의 총 지연을 고려해 봅시다. 이를 이해하기 위해, 소스 호스트와 목적지 호스트 사이에 N-1 개의 라우터가 있다고 가정합니다. 또한, 네트워크가 혼잡하지 않아 대기열 지연이 무시할 만하고, 각 라우터와 소스 호스트에서의 처리 지연이 d_proc, 각 라우터와 소스 호스트에서의 전송률이 R 비트/초, 각 링크의 전파 지연이 d_prop 라고 가정합시다. 노드 지연이 누적되어 종단 간 지연은 다음과 같습니다:

d_end-end = N * (d_proc + d_trans + d_prop)

여기서 d_trans = L/R 이고, L 은 패킷 크기입니다. 이 식은 처리 지연과 전파 지연을 고려하지 않은 식 1.1을 일반화한 것입니다. 각 노드에서 이종 지연과 평균 대기열 지연이 존재하는 경우로 식 1.2를 일반화하는 것은 독자들에게 연습 문제로 남겨두겠습니다.

Traceroute

컴퓨터 네트워크에서 종단 간 지연을 직접 체험하기 위해 Traceroute 프로그램을 사용할 수 있습니다. Traceroute는 모든 인터넷 호스트에서 실행할 수 있는 간단한 프로그램입니다. 사용자가 목적지 호스트 이름을 지정하면, 소스 호스트의 프로그램은 해당 목적지로 여러 개의 특수 패킷을 보냅니다. 이 패킷들은 목적지로 가는 도중에 여러 라우터를 통과합니다. 라우터가 이러한 특수 패킷 중 하나를 수신하면, 라우터의 이름과 주소를 포함한 짧은 메시지를 소스로 반환합니다.

더 구체적으로, 소스와 목적지 사이에 N-1 개의 라우터가 있다고 가정합시다. 소스는 최종 목적지로 주소가 지정된 N 개의 특수 패킷을 네트워크로 보냅니다. 이 N 개의 패킷은 1부터 N 까지 번호가 매겨지며, 첫 번째 패킷은 1, 마지막 패킷은 N 으로 표시됩니다. n 번째 라우터는 n 번으로 표시된 n 번째 패킷을 수신하면, 패킷을 목적지로 전달하지 않고 소스로 메시지를 반환합니다. 목적지 호스트는 N 번째 패킷을 수신하면 역시 소스로 메시지를 반환합니다. 소스는 패킷을 보내고 해당 반환 메시지를 수신하는 데 걸린 시간을 기록하며,

메시지를 반환한 라우터(또는 목적지 호스트)의 이름과 주소를 기록합니다. 이를 통해 소스는 소스에서 목적지로 흐르는 패킷의 경로를 재구성하고, 모든 중간 라우터로의 왕복 지연을 측정할 수 있습니다. Traceroute는 이 실험을 세 번 반복하므로, 소스는 실제로 3 * N 개의 패킷을 목적지로 보냅니다. RFC 1393은 Traceroute에 대해 자세히 설명합니다.

1.4.4 컴퓨터 네트워크의 처리량

지연과 패킷 손실 외에도 컴퓨터 네트워크에서 중요한 성능 지표는 종단 간 처리량입니다. 처리량을 정의하기 위해, 호스트 A에서 호스트 B로 큰 파일을 네트워크를 통해 전송하는 상황을 고려해 봅시다. 이 전송은 예를 들어 한 컴퓨터에서 다른 컴퓨터로 큰 비디오 클립을 전송하는 것일 수 있습니다. 순간 처리량은 특정 순간에 호스트 B가 파일을 수신하는 속도(비트/초)입니다. 파일이 F 비트로 구성되어 있고, 호스트 B가 모든 F 비트를 수신하는 데 T 초가 걸린다면, 파일 전송의 평균 처리량은 F/T 비트/초입니다.

두 링크 네트워크 (그림 1.19(a))

그림 1.19(a)는 서버와 클라이언트가 두 개의 통신 링크와 라우터로 연결된 두 개의 종단 시스템을 보여줍니다. 서버에서 클라이언트로 파일 전송의 처리량을 고려해 봅시다. 서버와 라우터 간 링크의 전송률을 R_s, 라우터와 클라이언트 간 링크의 전송률을 R_c 라고 합시다. 전체 네트워크에서 서버에서 클라이언트로 전송되는 비트만 있다고 가정합니다. 이 이상적인 시나리오에서 서버-클라이언트 처리량은 min{R_c, R_s}, 즉 병목 링크의 전송률입니다. 처리량을 결정한 후, F 비트의 큰 파일을 서버에서 클라이언트로 전송하는 데 걸리는 시간을 F/min{R_s, R_c} 로 근사할 수 있습니다.

구체적인 예시: F = 32 백만 비트(32Mb)의 MP3 파일을 다운로드한다고 가정합시다. 서버의 전송률은 R_s = 2 Mbps,bees 클라이언트의 접근 링크는 R_c = 1 Mbps입니다. 파일 전송에 필요한 시간은 다음과 같습니다:

Throughput = min{2, 1} = 1 Mbps

T = (32 * 10^6) / (1 * 10^6) = 32 초

물론, 이러한 처리량과 전송 시간 식은 저장-전달 지연, 처리 지연, 프로토콜 문제를 고려하지 않으므로 근사치일 뿐입니다.

N 개 링크 네트워크 (그림 1.19(b))

그림 1.19(b)는 서버와 클라이언트 사이에 N 개의 링크가 있는 네트워크를 보여주며, 각 링크의 전송률은 R_1, R_2, ..., R_N 입니다. 두 링크 네트워크와 동일한 분석을 적용하면, 서버에서 클라이언트로의 파일 전송 처리량은 다음과 같습니다:

Throughput = min{R_1, R_2, ..., R_N}

이는 다시 한 번 서버와 클라이언트 간 경로에서 병목 링크의 전송률입니다.

현대 인터넷 예시 (그림 1.20(a))

그림 1.20(a)는 서버와 클라이언트라는 두 개의 종단 시스템이 컴퓨터 네트워크에 연결된 모습을 보여줍니다. 서버는 R_s 의 전송률을 가진 접근 링크로, 클라이언트는 R_c 의 전송률을 가진 접근 링크로 네트워크에 연결되어 있습니다. 통신 네트워크 코어의 모든 링크는 R_s 와 R_c 보다 훨씬 높은 전송률을 가지며, 혼잡이 거의 없다고 가정합시다. 이 예시에서 컴퓨터 네트워크 코어는 넓은 파이프와 같으므로, 소스에서 목적지로 비트가 흐를 수 있는 속도는 다시 한 번 min{R_s, R_c} 입니다. 즉:

Throughput = min{R_s, R_c}

다중 다운로드 예시 **(**그림 **1.20(b))**

그림 1.20(b)를 고려해 봅시다. 여기서는 10개의 서버와 10개의 클라이언트가 컴퓨터 네트워크 코어에 연결되어 있습니다. 이 예시에서는 10개의 클라이언트-서버 쌍이 참여하는 10개의 동시 다운로드가 진행 중입니다. 현재 네트워크의 유일한 트래픽은 이 10개의 다운로드라고 가정합니다. 모든 10개의 다운로드가 통과하는 코어의 공통 링크가 있습니다. 이 링크의 전송률을 R 이라고 합시다.

모든 서버 접근 링크의 전송률이 R_s, 모든 클라이언트 접근 링크의 전송률이 R_c, 그리고 공통 링크를 제외한 코어의 모든 링크 전송률이 R_s, R_c, R 보다 훨씬 크다고 가정합시다. 공통 링크는 10개의 다운로드에 전송률을 균등히 나누므로, 각 다운로드는 다음과 같은 처리량을 가집니다:

Throughput per download = R/10

구체적인 예시:

- R_s = 2 Mbps, R_c = 1 Mbps, R = 5 Mbps.

Throughput per download = (5 * 10^6) / 10 = 500 kbps

이 경우, 각 다운로드의 병목은 더 이상 접근 네트워크(예: min{2, 1} = 1 Mbps)가 아니라, 이제 코어의 공유 링크로, 각 다운로드에 500kbps의 처리량만 제공합니다. 따라서 각 다운로드의 종단 간 처리량은 500kbps로 줄어듭니다.

처리량에 대한 주요 인사이트

그림 1.19와 그림 1.20(a)의 예시는 처리량이 데이터가 흐르는 링크의 전송률에 의존한다는 것을 보여줍니다. 다른 중간 트래픽이 없을 때, 처리량은 소스와 목적지 간 경로에서 최소 전송률로 간단히 근사할 수 있습니다. 그림 1.20(b)의 예시는 더 일반적으로, 처리량이 경로 상의 링크 전송률뿐만 아니라 중간 트래픽에도 의존한다는 것을 보여줍니다. 특히, 전송률이 높은 링크라도 많은 다른 데이터 흐름이 해당 링크를 통과하면 파일 전송의 병목 링크가 될 수 있습니다. 컴퓨터 네트워크의 처리량은 숙제 문제와 이후 장에서 더 자세히 살펴볼 것입니다.