구글 엔지니어에게 듣는

네트워킹과 웹 성능 최적화 기법

HIGH PERFORMANCE **BROWSER NETWORKING**

High Performance Browser Networking
by Ilya Grigorik

Authorized Korean translation of the English edition of HIGH PERFORMANCE BROWSER NETWORKING ISBN 9781149344764 ⓒ 2013 Ilya Grigorik

구글 엔지니어에게 듣는 네트워킹과 웹 성능 최적화 기법

전자책 1쇄 발행 2022년 12월 8일 **지은이** 일리아 그리고릭 **옮긴이** 정해권, 오현주 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트
편집 이지연 **등록번호** 제2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울시 마포구 연남로5길 19-5 **전화** 02-322-5143
팩스 02-3143-5579 **블로그** https://blog.insightbook.co.kr **이메일** insight@insightbook.co.kr **ISBN** 978-89-6626-383-7

ProgrammingInsight

구글 엔지니어에게 듣는

네트워킹과 웹 성능 최적화 기법

HIGH PERFORMANCE **BROWSER NETWORKING**

일리아 그리고릭 지음
정해권, 오현주 옮김

인사이트
insight

차례

옮긴이의 글

모바일 시대를 맞이하여 수많은 서비스, 제품들이 하루가 멀다 하고 홍수처럼 쏟아져 나오고 있습니다. 서비스, 제품을 만드는 기업은 사용자에게 좀 더 친근하게 다가가고 그들의 마음을 사로잡기 위해 다양한 노력을 하고 있습니다. 그 중에서 최근에 가장 주목하고 있는 분야는 사용자 경험(User Experience)입니다. 사용자 경험은 인간-컴퓨터 상호작용(HCI) 연구에서 쓰인 개념이며, 아직도 많은 사용자 경험의 원리가 컴퓨터공학 분야의 소프트웨어 및 하드웨어 개발에서 비롯되고 있습니다.

그렇다면 개발자, 엔지니어로서 사용자에게 좋은 사용자 경험을 제공하기 위해 할 수 있는 것은 무엇일까요? 그것은 같은 기능을 만들어도 '더 빠르게', '배터리 소모량은 더 적게' 다시 말해 '더 효율적으로' 만들어야 합니다.

이 책은 성능 개선에 관심 있는 웹 개발자, 모바일 앱 개발자, 서버 엔지니어 모두에게 꼭 필요한 네트워크 기본 원리와 개념, 성능을 끌어올릴 수 있는 테크닉에 대해 총 4부에 걸쳐 설명하고 있습니다.

1부에서는 네트워크 성능을 이해하기 위한 기본 용어 레이턴시와 대역폭에 대한 설명으로 시작되며, 가장 기본이 되는 핵심 프로토콜 TCP, UDP, TLS에 대해 철저히 파헤칩니다. 각 프로토콜별로 성능에 영향을 끼치는 요소에 대해 설명하고 성능을 최적화하는 방법에 대해 기술하고 있습니다.

2부에서는 모바일 환경에서 이루어지는 네트워크 기초 원리에 대해 설명하고 3G, 4G, 와이파이 환경별로 성능을 높이기 위해 알아야 할 특징에 대해 설명합니다.

3부에서는 웹 개발자, 앱 개발자들이 꼭 알아야 할 HTTP 프로토콜에 대해 완벽히 해부하고 HTTP의 간략한 역사와 함께 어떻게 HTTP 2.0까지 발전하게 되었는지 이야기 해 줍니다.

마지막으로 4부에서는 웹 개발자들이 서비스 성능 품질을 높이기 위한 브라우저 원리, 롱폴링 기법, 웹소켓 등 다양한 테크닉과 함께 여러 가지 브라우저 API에 대해 설명합니다.

구글을 비롯한 글로벌 소프트웨어 회사들은 이 책의 저자 일리아 그리고릭(Ilya

Grigorik)과 같은 성능 튜닝 엔지니어가 많습니다. 그들은 성능 튜닝과 최적화가 얼마나 중요한지 인식하고 있기 때문에 사용자에게 차원이 다른 서비스를 제공할 수 있는 것이라고 저는 확신합니다.

우리나라도 소프트웨어 강국으로 발돋움하기 위해 소프트웨어 설계의 자그마한 부분 하나하나가 사용자 경험 전반에 영향을 끼친다는 것을 깊이 인식하게 되길 바랍니다. 이 책에서 설명하고 있는 많은 개념과 노하우를 옆에 있는 동료, 팀원들과 함께 학습하고 토론하며 실력을 서로 배양해 나갔으면 합니다.

이 책을 번역할 수 있도록 기회를 주신 인사이트 한기성 사장님과 이지연 편집자님, 또한 번역에 있어 많은 도움을 주신 이복연 님께 깊이 감사드립니다.

끝으로 언제나 항상 저희를 응원해 주시는 부모님께 감사하다는 말을 전하고 싶습니다.

- 정해권, 오현주

추천사

"좋은 개발자는 시스템이 어떻게 동작하는지 안다. 그러나 훌륭한 개발자는 그것이 왜 그렇게 동작하는지를 이해한다"

우리는 이 명언에 모두 공감한다. 우리는 항상 사용하고 있는 시스템의 내부에 대해서도 샅샅이 이해하고 설명할 수 있는 사람이 되고 싶어 한다. 그럼에도 불구하고 당신이 웹 개발자라면 당신은 전혀 정 반대의 길로 가고 있는지도 모른다.

웹 개발의 영역은 나날이 전문화, 세분화 되어 가고 있다. 당신은 어떤 웹 개발자인가? 프론트엔드 영역, 백엔드 영역, Ops(데브옵스), 빅데이터 분석, UI/UX, 스토리지, 비디오 처리, 메시징 처리, 여기에 '퍼포먼스 엔지니어링 영역'까지 더한다면 웹 개발의 전문 분야는 더 늘어날 것이다.

기존 기술 스택의 기초에 대해 학습하는 것과 새로운 혁신 기술을 따라가는 것 사이에서 균형을 맞추기란 쉬운 일이 아니다. 하지만 기초에 대한 이해가 없다면 우리의 지식은 얕고 얄팍할 수밖에 없다. 기술 스택의 가장 상단의 사용방법에 대해 아는 것만으로는 턱없이 부족하다. 복잡한 문제를 해결해야 할 때, 혹은 이해하기 어려운 일이 벌어졌을 때, 기술의 근간 기초를 이해하고 있는 사람만이 문제를 해결하고 앞서나갈 수 있다.

이것이 왜 이 책이 중요한지를 말해준다. 당신이 웹 개발자라면 당신에게 맞는 기술 스택의 기초는 웹 기술과, 웹 기술의 기반이 되는 TCP, TLS, UDP, HTTP 프로토콜, 그리고 그 밖의 수많은 네트워킹 관련 프로토콜이다. 이 프로토콜들은 각각의 성능적인 특성과 최적화 방식이 다 다르기 때문에 고성능의 웹 애플리케이션을 구축하기 위해서는 네트워크가 동작하는 내부 원리에 대해 이해해야 할 필요가 있다.

당신이 이 책을 발견하게 된 것은 정말 감사한 일이다. 내가 웹 개발을 시작했을 때 이 책이 있었더라면 얼마나 좋았을까? 나의 경우 네트워크가 동작하는 원리를 이해하는 사람들로부터 직접 지식을 전수받고, 나머지는 프로토콜 스펙 문서를 읽어가며 이론과 실전의 차이를 줄여 나갔었다. 이 책은 네트워크 분야의 권위자 일리아 그리고릭(Ilya Grigorik)의 전문지식과 관련 기술 스펙 문서에서 추출한 꼭 필요한 정보들을 모두 한곳에 모아 놓은 책이다.

이 책에서 저자는 네트워킹의 원리를 많은 이유를 들어서 설명하고 있다. 왜 레이턴시가 성능의 병목이 되는가? 왜 TCP가 항상 최선의 전달 메커니즘이 아닌가? 그리고 UDP가 더 좋은 선택이 될 수 있지 않을까? 또한 왜 커넥션을 재사용하는 것이 최적화에 큰 도움이 되는가? 그리고 더 나아가서는 특정 네트워킹 성능을 향상시키기 위한 행동지침을 제시하기도 한다. 가령 레이턴시를 줄이고 싶으면 어떻게 해야 하는가? 클라이언트와 가까운 서버의 세션을 종료하라. 커넥션 재사용률을 늘리고 싶다면? 커넥션 킵얼라이브(Keep-alive) 옵션을 사용하라. 어떠한 문제에 대해서 어떻게 해야 할지를 이해하는 것과 왜 그것이 중요한지를 이해하는 능력이 결합되면 자신의 지식을 행동으로 옮길 수 있는 힘이 생긴다.

저자는 이 책에서 네트워킹의 기초에 대해 설명하는 동시에 그것을 기반으로 프로토콜과 브라우저의 최신 기술에 대해서 소개하고 있다. HTTP 2.0의 장점에 대해 설명하고 XHR(XMLHttpRequest)에 대해 살펴보며 XHR의 한계점을 극복하기 위해 교차 출처 자원 공유(Cross-Origin Resource Sharing, CORS) 메커니즘을 소개한다. SSE(Server-Sent Events), 웹소켓, WebRTC 등의 최신 브라우저 네트워킹 기술에 대해서도 다루고 있다.

성능의 관점에서 네트워킹의 기초와 최신 기술들을 바라보는 것이 이 책의 주요 특징이다. 성능은 네트워킹의 원리와 그것이 웹사이트와 유저들에게 어떠한 영향을 미치는지 알게 해 주는 척도가 된다. 이 책은 추상적이고 개괄적인 프로토콜 스펙의 내용들을, 당신이 개발하는 웹사이트를 최적화시키고 최고의 사용자 경험을 개발할 수 있도록 실질적 도구를 제공해 준다. 이것이 당신이 이 책을 읽어야 하는 이유다.

- 스티브 사우더(Steve Souders), 수석 웹 성능 엔지니어, 구글

서문

웹브라우저는 현재 개발자들에게 가장 널리 이용되고 있는 플랫폼이다. 웹브라우저는 모든 스마트폰, 태블릿, 노트북, 데스크탑, 그리고 그 밖의 많은 기기들에 설치되어 있다. 현재 IT 산업시장의 성장 추세를 보면 2020년에는 2백억 개의 네트워크 기기들이 존재할 것으로 보이며, 그 기기들은 모두 적어도 와이파이(WiFi)나 모바일 통신을 갖추고 있고 브라우저를 탑재하고 있을 것이다. 플랫폼의 종류나 기기를 제조하는 생산업체, 운영체제의 버전 따위는 중요치 않다. 중요한 것은 그 모든 기기가 웹브라우저를 탑재하고 있다는 사실이고, 날이 갈수록 웹브라우저의 기능은 점점 더 다양해지고 있다는 사실이다.

과거의 브라우저는 현재 우리가 접하고 있는 최신기술을 갖춘 브라우저의 모습과는 전혀 다른 모습이었다. HTML과 CSS의 프레젠테이션 레이어 언어와 자바스크립트가 웹상의 새로운 어셈블리 언어로 떠올랐다. 새로운 HTML5 API는 나날이 기능이 향상되고 있으며 고성능 애플리케이션의 탄생을 가능케 하는 새로운 플랫폼의 가능성을 보여주고 있다. 다른 어떤 기술이나 플랫폼도 현재 우리가 가진 브라우저와 같은 발전과 보급률을 가지지는 못했다. 커다란 기회가 있는 곳에 혁신이 따르기 마련이다.

사실 브라우저 내에서도 네트워킹 인프라만큼 급속한 발전과 혁신이 이루어진 예는 찾아보기 어렵다. 지금껏 네트워킹은 간단한 HTTP 요청-응답으로만 제한되어 있었으나 오늘날 우리는 몇 줄의 간단한 자바스크립트 코드만으로 효율적인 스트리밍, 양방향 실시간 커뮤니케이션, 커스텀 애플리케이션 프로토콜, 그리고 P2P 비디오 컨퍼런싱과 피어 간 데이터 전달까지 가능하게 되었다.

그 결과는 어떻게 되었을까? 수억 대의 네트워크 기기, 각종 온라인 서비스에 넘치는 유저기반, 고성능 웹 애플리케이션에 대한 높은 수요가 혁신의 결과다. 속도는 어떤 애플리케이션에 있어서는 많은 특성 중 하나에 불과할 수 있으나 또 다른 애플리케이션에게는 가장 중요한 특성이 될 수도 있다. 그리고 고성능의 웹 애플리케이션을 만들기 위해서는 브라우저와 네트워크가 어떻게 상호작용 하는지에 대한 탄탄한 기초 지식이 필요하다. 그것이 이 책의 주제다.

1부

네트워킹 기초

1장

High Performance **Browser Networking**

레이턴시와 대역폭 이해의 첫걸음

속도는 하나의 특징이다

요 몇 년 사이 웹 성능 최적화(Web Performance Optimization, WPO) 산업이 급속도로 떠오르면서 웹의 속도와 보다 빠른 사용자 경험(user experience)이 날로 중요해지고 이에 대한 사용자들의 수요가 점차 커져 가고 있다. 이러한 수요는 갈수록 빨라지는 인터넷 환경에서 그저 사용자들의 심리적 요인에서만 비롯된 것이 아니라, 현존하는 수많은 온라인 비즈니스의 최소한의 성능을 보장하기 위한, 실험 결과를 토대로 분석된 최소한의 요건이다.

- 웹사이트가 빠를수록 사용자를 더 오래 붙잡아 둘 수 있다.
- 웹사이트가 빠를수록 사용자의 재방문율을 향상시킨다.
- 웹사이트가 빠를수록 사용자의 구매 전환율을 향상시킨다.

간단히 말해서, 속도는 특징이다. 빠른 속도를 내기 위해서는 여러 가지 요소와 근본적인 제약사항에 대해 이해해야 한다. 이 장에서는 모든 네트워크 트래픽의 성능을 좌우하는 두 가지 중요한 요소, 즉 레이턴시와 대역폭에 대해 알아보겠다(그림 1-1).

레이턴시

패킷을 전송하는 곳에서부터 전달받는 곳까지 이동하는 데 걸리는 시간

대역폭

논리적인 혹은 물리적인 통신 경로의 최대 처리량

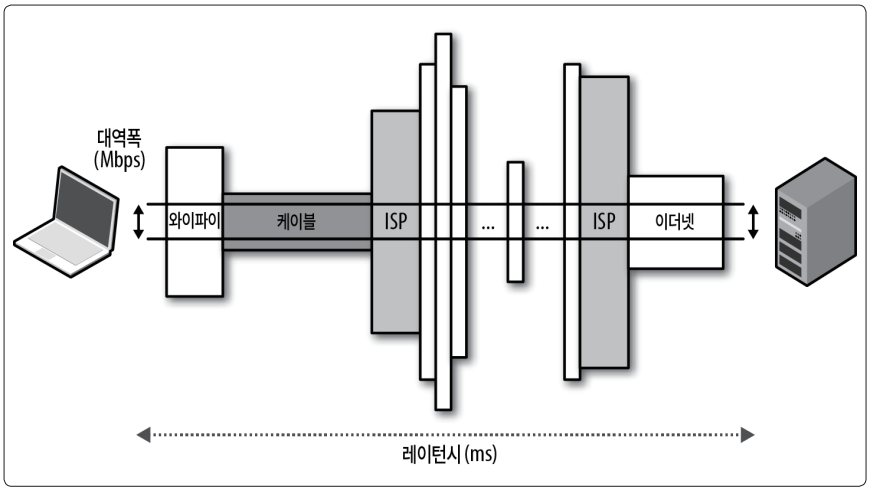

그림 1-1. 레이턴시와 대역폭

레이턴시와 대역폭이 어떤 식으로 함께 동작하는지를 이해하고 나면 TCP와 UDP, 그리고 그 위에 올라가는 모든 애플리케이션 프로토콜의 내부를 들여다 볼 것이며 그들의 고유한 성능을 이해하는 데 큰 도움이 될 것이다.

> **하이버니아 익스프레스(Hibernia Express)를 이용한 대서양 횡단의 레이턴시 줄이기**
>
> 레이턴시는 금융권에서 사용하는 거래 알고리즘의 중요한 기준이 된다. 천분의 일초 차이로 수백만 달러의 돈이 왔다 갔다 하기 때문이다.
>
> 2011년 초에 화웨이(Huawei)와 하이버니아 애틀랜틱(Hibernia Atlantic)이 'Hibernia Express'라는 프로젝트 이름으로 대서양에 런던과 뉴욕을 연결하는 4800여 킬로미터에 달하는 광섬유 케이블을 놓는 공사를 시작하였다. 이 작업의 유일한 목표는 런던과 뉴욕 사이에 존재하는 다른 어떤 대서양 연결 케이블보다 더 짧은 루트를 설치하여 증권가에서 투자자들의 레이턴시를 5밀리초만큼 단축하기 위함이었다.
>
> 이 케이블이 완성되면, 이것은 오직 금융권에서만 쓰일 예정이며 완공에 필요한 액수는 4억 달러에 달한다. 1밀리초를 단축시키는데 무려 8천만 달러를 쓰는 셈이다. 레이턴시는 상징적으로나 실질적으로나 매우 고비용이다.

레이턴시의 구성요소

레이턴시는 메시지나 패킷이 출발점에서 도착점까지 이동하는데 걸리는 시간을 말한다. 이러한 정의는 간단하고 유용한 정의이긴 하지만 그 속에 유용한 많은 정

보들을 감추고 있다. 모든 시스템은 메시지를 전송하는 데 걸리는 총 시간 안에 많은 소스와 구성 요소를 포함하고 있는데 이러한 구성 요소가 무엇인지, 그리고 시스템의 성능을 좌우하는 각각의 요소는 무엇인지 이해하는 것이 중요하다.

우선 인터넷 환경의 라우터에서 클라이언트와 서버 간 메시지를 주고받을 때 레이턴시를 일으키는 요소가 무엇인지 자세히 살펴보자.

전파 지연(propagation delay)

메시지가 송신측(Sender)에서 수신측(Receiver)으로 이동하는데 필요한 시간. 총 이동거리 대비 신호가 이동하는 속도로 측정된다.

전송 지연(transmission delay)

링크로 패킷의 모든 비트를 내보내는 데 필요한 시간. 패킷의 길이 대비 링크의 데이터 전송 속도로 측정된다.

프로세싱 지연(processing delay)

패킷 헤더를 처리하고 비트 수준(bit-level)의 에러를 체크하고 패킷의 목적지를 알아내는 데 필요한 시간.

큐잉 지연(queuing delay)

패킷이 처리될 때까지 큐(queue)에서 대기하는 시간

위에 기술된 4가지 지연을 합친 것이 클라이언트와 서버 간의 총 레이턴시다. 전파 지연은 신호가 이동하는 거리와 신호가 이동하는 데에 쓰이는 매체에 따라 달라진다. 곧 살펴보겠지만, 전파 속도는 보통 빛의 속도에서 크게 벗어나지 않는다. 반면 전송 지연은 전송 링크의 데이터 전송률에 좌우되며 클라이언트와 서버 간의 거리와는 아무런 상관이 없다. 예를 들어 우리가 1Mbps와 100Mbps 두 가지 링크를 이용해서 10Mb의 파일을 전송한다고 가정하자. 1Mbps 링크를 사용한다면 파일 전체를 보내는 데 10초가 걸리겠지만 100Mbps 링크를 사용하면 0.1초 밖에 걸리지 않는다.

그 후, 패킷이 라우터에 도착하면, 라우터는 다음 발송지 루트(outgoing route)를 알아내고 데이터를 체크하기 위해 패킷 헤더를 살펴봐야 한다. 이 과정 역시 시간이 소요된다. 이러한 작업들은 대부분 하드웨어에서 이루어지기 때문에 지연시간은 매우 적지만, 그래도 존재는 한다. 마지막으로, 패킷이 라우터가 한 번에 처리할 수 있는 속도보다 더 빠른 속도로 도착한다면 그 패킷들은 인커밍 버퍼

(incoming buffer) 안에서 잠시 대기하게 된다. 데이터가 이 버퍼 안에서 보내는 시간이 바로 큐잉 지연이다.

네트워크를 이동하는 각각의 패킷은 모두 이러한 지연을 겪게 된다. 데이터의 이동 거리가 멀수록 전파 지연은 늘어나고 중간 단계에서 라우터를 만날 때마다 각 패킷은 프로세싱 지연과 전송 지연을 겪게 된다. 마지막으로 이동 중에 트래픽이 높을수록 패킷이 인커밍 버퍼에서 지연될 확률도 높아진다.

> **로컬 라우터 안의 버퍼블롯**
>
> 버퍼블롯(Bufferbloat)이란 2010년 짐 게티즈(Jim Gettys)가 처음 만들어 사용한 용어로, 큐잉 지연이 네트워크 성능 전반에 영향을 끼칠 수 있다는 좋은 예다.
>
> 현재 판매되고 있는 많은 라우터는 커다란 인커밍 버퍼를 가지고 있는데, 이는 패킷을 최대한 손실하지 않기 위함이다. 그러나 이것이 다음 장에서 알아볼 TCP의 혼잡회피 메커니즘(congestion avoidance mechanism)을 오작동하게 만들어 네트워크에 크고 작은 레이턴시 지연을 야기한다. 다행히도 이러한 문제점을 해결하기 위해 코들 활성 큐 관리 알고리즘(CoDel active queue management algorithm)이 소개되어 Linux 3.5+ kernel에서 현재 구현되어 있다. 이에 대해 더 자세히 알고 싶다면 ACM Queue의 <Controlling Queue Delay>를 찾아보라.(http://hpbn.co/aqmacm)

빛의 속도와 전파 지연

아인슈타인이 상대성 이론에서 언급했듯이, 빛의 속도는 모든 에너지와 물질 그리고 정보가 이동할 수 있는 최고 속도다. 이는 모든 네트워크 패킷의 전파 시간을 엄격히 제한하고 있다.

다행히 빛의 속도는 매우 빠르다. 초당 299,792,458미터에 이른다. 하지만 이것은 진공 상태였을 때의 속도이고, 보통 패킷을 보낼 때 사용하는 매체는 구리선이나 광섬유 케이블이기 때문에 그만큼 신호의 속도는 느려지게 된다(표 1-1). 빛의 속도 대비 특정 물질 안에서 패킷이 이동하는 속도를 측정한 것을 물질의 굴절률(refractive index)이라 한다. 이 값이 클수록 해당 물질 안에서 빛이 이동하는 속도는 감소한다.

우리가 패킷을 장거리에 걸쳐 보낼 때 사용하는 광섬유의 보편적인 굴절률은 1.4에서 1.6 사이다. 우리는 계속해서 광섬유의 굴절률 값을 줄여 나가기 위해 꾸준히 노력하고 있지만, 간단하게 어림잡아 광섬유의 굴절률은 1.5 혹은 그에 약간

못 미치는 정도라고 생각하는 것이 옳다. 즉, 광섬유 안에서의 빛의 속도는 초당 약 2억 미터라고 보면 된다. 놀랍게도 이 속도는 이미 빛의 속도와 매우 근접해 있으며, 그 자체만으로도 굉장한 기술적 쾌거라 할 수 있다.

빛의 속도가 빠르다고는 하나 뉴욕에서 시드니까지의 왕복에만 160밀리초가 걸린다. 사실 표 1-1의 수치도 패킷이 두 도시를 최단거리로 연결하는 광섬유를 통해 이동한다는 가정하에 계산된 값이므로 현실적이지 않은 이상적인 값이다. 현실세계에서는 그러한 케이블이 존재하지 않기 때문에 패킷은 뉴욕에서 시드니까지 훨씬 긴 경로를 통해 이동하게 될 것이다. 패킷이 이동하면서 경로를 갈아탈 때마다 앞에서 설명한 라우팅, 프로세싱, 큐잉, 전송 지연을 얻게 된다. 결과적으로 실제 뉴욕과 시드니에 존재하는 네트워크를 왕복하는 데는 200-300밀리초가 걸리게 된다. 이 모든 것을 다 고려해도 여전히 꽤 빠르지 않은가?

경로	거리	시간(진공)	시간(광섬유)	왕복시간(광섬유)
뉴욕→샌프란시스코	4,148km	14ms	**21ms**	42ms
뉴욕→런던	5,585km	19ms	**28ms**	56ms
뉴욕→시드니	15,993km	53ms	**80ms**	160ms
적도의 둘레	40,075km	133.7ms	**200ms**	200ms

표 1-1. 진공 상태와 광섬유 내에서의 신호 레이턴시

우리는 보통 일상적인 일들을 밀리초 단위로 측정하지는 않지만, 연구결과에 따르면 대부분의 사람들은 시스템에 100-200밀리초 정도의 지연이 생기는 것을 꽤 정확하게 감지한다고 한다. 지연이 300밀리초 이상으로 넘어서면 시스템을 '느리다'라고 인식하고, 1초의 경계를 넘어가게 되면 많은 사용자들은 기다리는 중에 머릿속으로 딴 생각을 하거나 그 다음에 해야 할 일들에 대해서 생각하기 시작한다고 한다.

결론은 간단하다. 사용자들을 잘 붙잡아 두는 동시에 최고의 사용자 경험을 제공하기 위해서 우리의 애플리케이션은 수백 밀리초 이내에 응답할 수 있어야 한다. 특히 네트워크상에서는 오류를 허용할 여유가 전혀 없다. 성공적인 통신을 위해서는 네트워크 레이턴시를 주의 깊게 관리하고 모든 개발 단계에 있어서 이를 중요한 디자인 기준으로서 다뤄야 한다.

Content delivery network(CDN) 서비스는 많은 이점이 있지만 그 중에 가장 대표적인 것은 모든 데이터 패킷의 전파 시간을 대폭 줄일 수 있다는 점이다. 다시 말해, CDN이 지구 곳곳으로 콘텐츠를 배분하여 클라이언트와 가장 가까운 지점에서 전달받도록 하는 것이다.

우리는 패킷을 한 번에 더 빠르게 이동하게 할 수는 없을지라도 사용자와 가까운 곳에 서버를 위치시킴으로써 이동 거리를 줄일 수 있다. CDN을 잘 활용하면 데이터 전달 성능을 크게 향상시킬 수 있다.

최종 마일 레이턴시(Last-Mile Latency)

얄궂게도 대부분의 레이턴시가 발생하는 곳은 바다나 대륙을 건너는 도중이 아니라 마지막 몇 킬로미터 지점에서 발생한다. 집이나 사무실을 인터넷에 연결하기 위해서는 ISP가 각 지역에 케이블을 설치하고 신호를 취합하여 로컬 라우팅 노드로 넘겨주어야 한다. 네트워크의 연결방식, 라우팅 방법, 사용되는 기술 등에 의해 ISP의 메인 라우터에 도달하는 데에만 수십 밀리초가 걸릴 수도 있다. 2013년 초 연방 통신 위원회에서 행해진 "Measuring Broadband America" 보고서에 따르면, 가장 트래픽이 많은 시간대에는,

> 평균적으로 파이버 투 더 홈(Fiber-to-the-home, FTTH)은 레이턴시 18ms로 가장 높은 성능을 보였고 케이블은 26ms, DSL은 44ms의 레이턴시가 발생했다.
>
> - 연방 통신 위원회(FCC), 2013 2월

이 수치들은 ISP의 코어 네트워크(Core Network) 내에 존재하는 가장 가까운 지점에서 측정한 레이턴시 값이므로 패킷이 도착지점으로 라우팅 되기도 전의 수치다. 연방 통신 위원회 보고서는 미국의 상황에 초점이 맞춰져 있지만 최종 마일 레이턴시는 미국뿐 아니라 전세계의 모든 인터넷 제공 업체들이 풀어나가야 할 숙제다. 당신이 사용하고 있는 인터넷 제공업체의 토폴로지와 성능에 대해 자세히 알고 싶다면 traceroute라는 간단한 명령어로도 많은 정보를 얻을 수 있다.

```
$> traceroute google.com
traceroute to google.com (74.125.224.102), 64 hops max, 52 byte packets
 1 10.1.10.1 (10.1.10.1) 7.120 ms 8.925 ms 1.199 ms ❶
 2 96.157.100.1 (96.157.100.1) 20.894 ms 32.138 ms 28.928 ms
 3 x.santaclara.xxxx.com (68.85.191.29) 9.953 ms 11.359 ms 9.686 ms
 4 x.oakland.xxx.com (68.86.143.98) 24.013 ms 21.423 ms 19.594 ms
 5 68.86.91.205 (68.86.91.205) 16.578 ms 71.938 ms 36.496 ms
 6 x.sanjose.ca.xxx.com (68.86.85.78) 17.135 ms 17.978 ms 22.870 ms
 7 x.529bryant.xxx.com (68.86.87.142) 25.568 ms 22.865 ms 23.392 ms
 8 66.208.228.226 (66.208.228.226) 40.582 ms 16.058 ms 15.629 ms
 9 72.14.232.136 (72.14.232.136) 20.149 ms 20.210 ms 18.020 ms
```

```
10 64.233.174.109 (64.233.174.109) 63.946 ms 18.995 ms 18.150 ms
11 x.1e100.net (74.125.224.102) 18.467 ms 17.839 ms 17.958 ms ❷
```

❶ 첫 번째 홉: 로컬 무선 라우터

❷ 열한 번째 홉: 구글 서버

앞의 예에서 보면, 패킷은 써니베일(Sunnyvale)시에서 시작하여 산타클라라(Santa Clara)와 오클랜드(Oakland)를 거쳐 산호세(San Jose)로 돌아왔고, 그 이후 '529 Bryant' 데이터센터로 라우팅된 후에 다시 구글로 라우팅되어 11번의 홉 끝에 도착점으로 전달되었다. 이 모든 과정이 평균 18밀리초가 소요되었다. 모든 상황을 고려했을 때 나쁘지 않은 수치이지만, 이 시간은 사실 패킷이 미대륙 전체를 여행할 수도 있는 시간이다.

최종 마일 레이턴시는 당신이 사용하는 인터넷 제공업체, 사용되는 기술, 네트워크의 토폴로지, 심지어는 하루의 시간대에 따라서도 바뀔 수 있다. 인터넷 사용자로서 웹 브라우징의 속도를 높이고 싶다면 ISP를 선택할 때 레이턴시를 낮추는 데 중점을 두는 것이 좋다.

대부분의 웹사이트들에서 성능의 병목점은 대역폭이 아니라 레이턴시다. 그 이유를 이해하기 위해서는 다음 장에서 다루게 될 TCP와 HTTP 프로토콜이 어떻게 동작하는지 이해해야 한다. 궁금한 사람들은 190쪽의 "대역폭을 늘려도 큰 영향은 없다"를 미리 읽어보도록 하라.

Traceroute를 이용하여 레이턴시 측정하기

Traceroute는 IP 네트워크에서 발생하는 홉의 레이턴시와 패킷의 이동경로를 파악할 수 있는 간단한 네트워크 진단 도구다. Traceroute는 각 홉을 구분하기 위해서 '홉 한도'(1,2,3 등등)를 정해 일련의 패킷을 도착지점으로 보낸다. 패킷이 홉 한계에 다다르면 중계지점에서 ICMP 시간 초과 메시지를 보내옴으로써 각 네트워크 홉마다 발생하는 레이턴시를 측정할 수 있게 한다.

유닉스 플랫폼에서는 traceroute 명령어를 입력하여 사용할 수 있고 윈도우(Windows)에서는 tracert 명령어를 사용한다.

코어 네트워크(Core Networks)의 대역폭

광섬유는 사람의 머리카락보다 약간 더 두꺼운 '빛을 뿜는 파이프'로써, 케이블 한 쪽 끝에서 다른 한쪽 끝까지 빛을 전달하도록 설계되었다. 광섬유뿐만 아니라 금속 전선도 연결에 사용되기도 하지만 금속 전선은 신호 손실이 더 많고 방해 전파의 영향을 더 많이 받으며 유지비용도 더 많이 든다. 패킷은 광섬유 케이블과 금

속 케이블 양쪽 모두를 사용하여 이동하지만 장거리 홉에서는 광섬유 링크를 통해 전달될 것이다.

광섬유는 파장 분할 다중화(wavelength-division multiplexing, WDM)를 통해 각 섬유마다 다른 파장의 빛을 이동시킬 수 있으므로 대역폭에 있어서는 확실한 이점을 가지고 있다. 그러므로 광섬유 링크의 총 대역폭은 다중 송신되는 채널의 수에 채널별 데이터 전송률을 곱한 만큼의 값이다.

2010년 초에 연구진들은 광섬유 링크를 이용하여 채널당 최대 171Gbit/s로 400개 이상의 채널을 다중화 할 수 있게 되었다. 하나의 광섬유 링크당 70Tbit/s의 대역폭을 갖게 되는 것이다. 만약 이만큼의 처리량을 구리전선으로 구성한다면 전선이 수천 개는 필요할 것이다. 그러므로 당연히 대륙을 연결하는 해저 데이터 송수신과 같은 장거리 홉은 모두 광섬유 링크에서 이루어지고 있다. 각 케이블은 몇 가닥의 섬유(보통 4가닥으로 이루어져 있음)로 구성되어 있는데, 이는 각 케이블마다 초당 수백 테라 비트의 대역폭으로 해석할 수 있다.

네트워크 가장자리(Network Edge)에서의 대역폭

인터넷의 핵심 데이터 패스를 구성하는 백본망 혹은 광섬유 링크는 초당 수백 테라 비트의 데이터를 옮길 수 있다. 하지만 네트워크의 끝자락에서는 사용하는 기술에 따라서 그 허용치는 훨씬 적다. 전화선, DSL, 케이블 그리고 여러 무선 기술들, 파이버 투 더 홈(fiber-to-the-home), 심지어 가정용 라우터의 성능도 대역폭에 영향을 미친다. 사용자에게 주어지는 대역폭은 클라이언트와 목적지점의 서버 간에 존재하는 링크 중 가장 낮은 허용치와 같다(그림 1-1).

아카마이 테크놀로지사(Akamai Technologies)는 세계 각국에 서버를 두고 글로벌 CDN을 운영하고 있으며 아카마이 웹사이트에 분기별로 각 서버의 평균 광대역(broadband) 속도를 기록한 보고서를 공개하고 있다. 표 1-2는 2013년 1분기의 거시적인 대역폭 트렌드를 나타내고 있다.

이 데이터에서는 이동통신 사업자에서 오는 트래픽은 이후 장에서 더 알아볼 것임으로 일단 여기서는 제외하였다. 일단 여기서는 무선 통신의 속도는 일반적으로 더 느리고 상당히 가변적이라는 것만 알아두면 된다. 하지만 이점을 고려하더라도 2013년 초의 전세계 평균 광대역(broadband) 대역폭은 3.1Mbps였다. 대한민국이 평균 처리량 14.2Mbps로 가장 앞서나가고 미국은 8.6Mbps로 9위에 머물렀다.