Networked Life

How does Google sell ad space and rank webpages? How does Netflix recommend movies, and Amazon rank products? How can you influence people on Facebook and Twitter, and can you really reach anyone in six steps? Why doesn't the Internet collapse under congestion, and does it have an Achilles' heel? Why are you charged per gigabyte for mobile data, and how can Skype and BitTorrent be free? How big is the "cloud" of cloud services, and why is WiFi slower at hotspots than at home?

Driven by 20 real-world questions about our networked lives, this book explores the technology behind the multi-trillion dollar Internet, wireless and online media industries. Providing easily understandable answers for the casually curious, alongside detailed explanations for those looking for in-depth discussion, this thought-provoking book is essential reading for students in engineering, science and economics, for network industry professionals, and for anyone curious about how technological and social networks really work.

Mung Chiang is a Professor of Electrical Engineering at Princeton University, and Director of the Princeton EDGE Lab. He has received the IEEE Kiyo Tomiyasu Award, and a US Presidential Early Career Award for Scientists and Engineers, for his research on networking. A co-founder and advisor to several startups, he also received a Technology Review TR35 Award for his contributions to network technology innovation. He is a fellow of the IEEE.

“We are entering a new Internet era – the era of the likes of Google, Amazon, Netflix, and Facebook – with entirely new types of problems. This book captures the new era, taking a fresh approach to both topic coverage and pedagogic style. Often at the end of a section it leaves the reader asking questions; then exactly those questions are answered in the subsequent section. Every university should offer a course based on this book. It could be taught out of both ECE or CS departments at the undergraduate or graduate levels.”

Keith Ross, Polytechnic Institute of NYU

“How do the networks, which we increasingly rely upon in our everyday life, actually *work*? This book is an inspiring romp through the big ideas in networking, which is immediately rewarding and will motivate later courses.”

Frank Kelly, University of Cambridge

Networked Life

20 Questions and Answers

MUNG CHIANG
Princeton University

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9781107024946

First published 2012

Printed in the United States by Edwards Brothers

A catalog record for this publication is available from the British Library

ISBN 978-1-107-02494-6 Hardback

Additional resources for this publication at www.cambridge.org/networkedlife

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

To my family

Contents

Preface *page* ix

Acknowledgements xii

Roadmap xv

1 **What makes CDMA work for my smartphone?** 1

2 **How does Google sell ad spaces?** 25

3 **How does Google rank webpages?** 44

4 **How does Netflix recommend movies?** 61

5 **When can I trust an average rating on Amazon?** 89

6 **Why does Wikipedia even work?** 110

7 **How do I viralize a YouTube video and tip a Groupon deal?** 129

8 **How do I influence people on Facebook and Twitter?** 158

9 **Can I really reach anyone in six steps?** 194

10 **Does the Internet have an Achilles' heel?** 214

11 **Why do AT&T and Verizon Wireless charge me $10 a GB?** 235

12 **How can I pay less for each GB?** 256

13 **How does traffic get through the Internet?** 277

14 **Why doesn't the Internet collapse under congestion?** 309

15 **How can Skype and BitTorrent be free?** 334

16 **What's inside the cloud of iCloud?** 358

17 **IPTV and Netflix: How can the Internet support video?** 380

18 **Why is WiFi faster at home than at a hotspot?** 406

19 **Why am I getting only a few % of the advertised 4G speed?** 433

20 **Is it fair that my neighbor's iPad downloads faster?** 452

Index 473

Notes 479

Preface

You pick up your iPhone while waiting in line at a coffee shop. You Google a not-so-famous actor and get linked to a Wikipedia entry listing his recent movies and popular YouTube clips. You check out user reviews on IMDb and pick one, download that movie on BitTorrent or stream it in Netflix. But for some reason the WiFi logo on your phone is gone and you're on 3G. Video quality starts to degrade a little, but you don't know whether it's the video server getting crowded in the cloud or the Internet is congested somewhere. In any case, it costs you $10 per gigabyte, and you decide to stop watching the movie, and instead multitask between sending tweets and calling your friend on Skype, while songs stream from iCloud to your phone. You're happy with the call quality, but get a little irritated when you see that you have no new followers on Twitter.

You've got a typical networked life, an online networked life.

And you might wonder how all these technologies "kind of" work, and why sometimes they don't. Just flip through the table of contents of this book. It's a mixture: some of these questions have well-defined formulations and clear answers while for others there is still a significant gap between the theoretical models and actual practice; a few don't even have widely accepted problem statements. This book is about formulating and answering these 20 questions.

This book is about the networking technologies we use each day as well as the fundamental ideas in the study of networks. Each question is selected not just for its relevance to our daily lives, but also for the core concepts and key methodologies in the field of networking that are illustrated by its answer. These concepts include aggregation and influence, distributed coordination, feedback control, and strategic equilibrium. And the analytic machineries are based on mathematical languages that people refer to as graph, optimization, game, and learning theories.

This is an undergraduate textbook for a new course created in 2011 at Princeton University: **Networks: Friends, Money, and Bytes**. The course targets primarily juniors and seniors in electrical engineering and computer science, but also beginning graduate students as well as students from mathematics, sciences, economics, and engineering in general. It can be viewed as a second course after the "signals and systems" course that anchors the undergraduate electrical and computer engineering curriculum today. Starting in September 2012, this course is also on free open access platforms, such as Stanford's coursera and the course's own open education website, as well as on YouTube and iTunes U.

This book weaves a diverse set of topics you would not normally see under the same cover into a coherent stream: from Arrow's impossibility and Rawls' fairness to Skype signaling and Clos networks, from collaborative filtering and firefly synchronization to MPEG/RTSP/TCP/IP and WiFi CSMA DCF. This begs a question: "So, what *is* the discipline of this book?" This is a question that most of the undergraduates do not care about. Neither does this book, which only wants to address these practical questions, using whatever modeling languages that have been observed to be the most relevant ones so far. It turns out that there is a small, common set of mathematics which we will need, but that's mostly because people have invented only a limited suite of modeling languages.

This is not a typical textbook for another reason. It does not start with general theories as do many books on these subjects, e.g., graph theory, game theory, and optimization theory, or with abstract concepts like feedback, coordination, and equilibrium. Instead it starts with concrete applications and practical answers, and sticks to them (almost) every step of the way. Theories and generalizations emerge, as if they were "accidental by-products," during the process of formulating and answering these questions.

This book can be supplemented with its website: `http://www.network20q.com`, including lecture slides, problem solutions, additional questions, examples of advanced material, further pointers to references, collections of news media coverage of the topics, "currency-earning" activities, course projects, blogs, tweets, surveys, and student-generated course material in wiki. We have created web features that turn this class into an online social network and a "networked economy."

This book can also be used by engineers, technology managers, and pretty much anyone with a keen interest in understanding how social and technological networks work. Often we sacrifice generality for accessibility, and supplement symbolic representation with numerical illustration.

- The first section of each chapter is a "short answer," and it is accessible by most people.
- Then there's a "long answer" section. If you remember differentiation and linear algebra (and occasionally a little bit of integration and basic probability), you can follow all the material there. We try to include only those symbols and equations that are really necessary to unambiguously express the ideas.
- The "examples" section contains detailed, numerical examples to reinforce the learning from the "long answer" section. On average, these first three sections of each chapter form the basis of one 80-minute lecture. Several of these lectures will go over 80 minutes while several others, including the last two, can be covered under 80 minutes.

- Each chapter concludes with a section on "advanced material," which requires the reader to be quite comfortable with symbolic operations and abstract reasoning, but can be skipped without losing the coherence and gist of the book. In the undergraduate course taught at Princeton, hardly any of the advanced material is covered. Covering all the "advanced material" sections would constitute an introductory graduate-level course. To keep the book thin, worked examples for these sections are pushed to the course website.
- At the end of each chapter, there are five homework questions, including easy drills, essential supplements, and some "out-of-syllabus" explorations about networks in biology, energy, magic, music, and transportation. The level of difficulty is indicated on a scale of one (easy) to three (hard) stars. On the course website, there is a much larger collection of additional homework problems, including many multiple-choice questions testing the basic understanding of the material.
- There are also five key references per chapter (yes, only five, in the hope that undergraduates may actually read some of these five, and my apologies to the authors of thousands of papers and books that could have been cited). These references open the door to further reading, including textbooks, research monographs, and survey articles.

This is a (relatively) thin book. It's a collage of snapshots, not an encyclopedia. It's an appetizer, not an entree. The majority of readers will not pursue a career specializing in the technical material in this book, so I take every opportunity to delete material that's very interesting to researchers but not essential to this undergraduate course. Each one of these 20 chapters deserves many books for a detailed treatment. I only highlight a few key ideas in the span of about 20 pages per chapter and 80 minutes per lecture. There are also many other mathematical languages in the study of networks, many other questions about a networked life, and many other types of networks that we do not have time to cover in one semester. But as the saying goes for a course: "It's more important to *uncover* than to cover a lot."

This is a book illustrating some pretty big ideas in networking, through 20 questions we can all relate to in our daily lives. Questions that tickle our imagination with surprises and incomplete answers. Questions that I wished I had known how to answer several years ago. Questions that are quickly becoming an essential part of modern education in electrical and computer engineering.

But above all, I hope this book is fun to read.

Mung Chiang
Princeton, NJ
July 2012

Acknowledgements

In so many ways I've been enjoying the process of writing this book and creating the new undergraduate course at Princeton University. The best part is that I got to, ironically in light of the content of this book, stay *offline* and focus on learning a few hours a day for several hundred days. I got to digest wonderful books and papers that I didn't have a chance to read before, to think about the essential points and simple structures behind the drowning sea of knowledge in my research fields, and to edit and re-edit each sentence I put down on paper. It reminded me of my own sophomore year at Stanford University one and a half decades ago. I often biked to the surreally beautiful Oval in the morning and dived into books of many kinds, most of which were not remotely related to my majors. As the saying goes, that was a pretty good approximation of paradise.

That paradise usually ends together with the college years. So I have many people to thank for granting me a precious opportunity to indulge myself again at this much later stage in life.

- The new course "Networks: Friends, Money, and Bytes" could not have been created in 2011 without the dedication of its three remarkable TAs: Jiasi Chen, Felix Wong, and Pei-yuan Wu. They did so much more for the course than a "normal" TA experience: creating examples and homework problems, initiating class activities, and running the online forum.
- Many students and postdocs in Princeton's EDGE Lab and EE Department worked with me in creating worked examples and proofreading the drafts: Chris Brinton, Amitabha Ghosh, Sangtae Ha, Joe Jiang, Carlee Joe-Wong, Yiannis Kamitsos, Haris Kremo, Chris Leberknight, Srinivas Narayana, Soumya Sen, Victoria Solomon, Arvid Wang, and Michael Wang.
- Princeton students in ELE/COS 381's first offering were brave enough to take a completely new course and contributed in many ways, not the least through the class website blogs and course projects. Students in the graduate course ELE539A also helped proofread the book draft and created multiple choice questions.
- Before I even got a chance to advertise the course, some colleagues already started planning to offer this course at their institutions in 2012: Jianwei Huang (Chinese University of Hong Kong), Hongseok Kim (Sogang University, Korea), Tian Lan (George Washington University), Walid Saad

(University of Miami), Chee Wei Tan (City University of Hong Kong), and Kevin Tang (Cornell University).

- The material in this book is inspired by discussions with colleagues in both academia and industry over the years. Since last summer, more than fifty colleagues has provided valuable suggestions directly for the course and the book. In particular, I received very detailed comments on earlier drafts of the book from Keith Cambron (AT&T Labs), Kaiser Fung (Sirius), Victor Glass (NECA), Jason Li (IAI), Jennifer Rexford (Princeton), Keith Ross (NYU Poly), Krishan Sabnani (Bell Labs), Walid Saad (University of Miami), Matthew Salganik (Princeton), Jacob Shapiro (Princeton), Kevin Tang (Cornell), and Walter Willinger (AT&T Labs), among others.
- Phil Meyler from Cambridge University Press encouraged me to turn the lecture notes into a textbook, and further connected me with a group of enthusiastic staff at CUP. I am grateful to the entire Cambridge editorial and marketing team across their UK and New York offices.
- This course was in part supported by a grant from the US National Science Foundation, in a program run by Darleen Fisher, for a team consisting of two engineers and two social scientists at Princeton. I'm glad to report that we achieved this educational goal in our proposal, and did that before the project's official end date.

My appreciation traces back to many of my teachers. For example, I've had the fortune to be co-advised in my Ph.D. study by Stephen Boyd and Tom Cover, two brilliant scholars who are also superb teachers. Their graduate-level textbooks, *Convex Optimization* by Boyd and Vandenberghe and *Elements of Information Theory* by Cover and Thomas, are two towering achievements in engineering education. Read these two books, and you'll "experience" the definition of "clarity," "accessibility," and "insight." When I was writing research papers with them, Tom would spend many iterations just to get one notation right, and Stephen would even pick out each and every LaTex inconsistency. It was a privilege to see first-hand how the masters established the benchmarks of technical writing.

Stephen and Tom were also the most effective lecturers in classroom, as was Paul Cohen, from whom I took a math course in my sophomore year. Pulling off the sweatshirt and writing with passion on the blackboard from the first moment he entered the classroom, Paul could put your breath on hold for 80 minutes. Even better, he forgot to give us a midterm and then gave a week-long, take-home final that the whole class couldn't solve. He made himself available for office hours on-demand to talk about pretty much anything related to math. The course was supposed to be on PDE. He spent just four lectures on that, and then introduced us to eighteen different topics that quarter. I've forgotten most of what I learned in that course, but I'll always remember that learning can be so much fun.

In the same winter quarter that I took Stephen's and Tom's courses, I also took from Richard Rorty a unique course at Stanford called "From Religion through Philosophy to Literature," which pulled me out of Platonism to which I had been increasingly attached as a teenager. Talking to Rorty drastically sharpened my appreciation of the pitfalls of mistaking representations for reality. A side-benefit of that awakening was a repositioning of my philosophy of science, which propagated to the undercurrents of this book.

Three more inspirations, from those I never met:

- Out of all the biographies I've read, the shortest one, by far, is by Paul Johnson on Churchill. And it's by far the most impactful one. Brevity is power.
- But even a short book feels infinitely long to the author until it goes to the press. What prevented me from getting paralyzed by procrastination is Frederick Terman's approach of writing textbooks while leading a much busier life (serving as a Dean and then the Provost at Stanford, and creating the whole Silicon Valley model): write one page each day.
- Almost exactly one century ago, my great grandfather, together with his brother, wrote some of the first modern textbooks in China on algebra and on astronomy. (And three decades ago, my grandfather wrote a textbook on econometrics at the age of seventy.) As I was writing this book, sometimes I couldn't help but picture the days and nights that they spent writing theirs.

For some reason, the many time commitments of a professor are often hard to compress. And I couldn't afford to cut back on sleep too much, for otherwise the number of mistakes and typos in this book would have been even larger. So it's probably fair to say that each hour I spent writing this book has been an hour of family time lost. Has that been a good tradeoff? Definitely not. So I'm glad that the book is done, and I'm grateful to my family for making that happen: my parents who helped take care of my toddler daughter when I was off to dwell in this book, my wife who supported me sitting there staring at my study's ceiling despite her more important job of curing the ill, and Novia who could have played with her Daddy a lot more in the past year. This book was written with my pen and their time.

Roadmap

This roadmap is written for course instructors, or perhaps as an *epilogue* for students who have already finished reading the book, especially Figure 0.3 and the list of 20 ideas below it. It starts with a taxonomy of the book and introduces its organization and notation. Then it discusses the similarities and differences between this book and some excellent books published over the last decade. Then it highlights three pedagogical principles guiding the book: Just In Time, Bridge Theory and Practice, and Book As a Network, and two contexts: the importance of domain-specific functionalities in network science and the need for undergraduate-curriculum evolution in electrical and computer engineering. It concludes with anecdotes of arranging this course as a social and economic network itself.

Taxonomy and Organization

The target audience of this book is both students and engineering professionals. For students, the primary audience are those from engineering, science, economics, operations research, and applied mathematics, but also those on the quantitative side of sociology and psychology.

There are three ways to use this book as a textbook.

- *An undergraduate general course at sophomore or junior level*: Go through all 20 chapters without reading the Advanced Material sections. This course serves as an introduction to networks before going further into senior-level courses in four possible directions: computer networking, wireless communication, social networks, or network economics.
- *An undergraduate specialized course at senior level*: Pick either the social and economic network chapters or the technology and economic network chapters, and go through the Advanced Material sections in those chapters.
- *A first-year graduate level course*: Go through all 20 chapters, including the Advanced Material sections.

While this book consists of 20 chapters, there are just four key recurring concepts underlying this array of topics. Table 0.1 summarizes the mapping from chapter number to the concept it illustrates.

Table 0.1 Key concepts: The chapters where each of the four key concepts show up for different types of networks.

Network Type	Aggregation & Influence	Distributed Coordination	Feedback Control	Strategic Equilibrium
Wireless		1	19	
Internet		10, 13, 16	14	
Content Distribution		15, 17	18	
Web	3, 4, 5			2
Online Social	6,8	9	7	
Internet Economics		20		11, 12

The modeling languages and analysis machineries originate from quite a few fields in applied mathematics, especially the four foundations summarized in Table 0.2.

Table 0.2 Main methodologies: The chapters where each of the four families of mathematical languages are used in different types of networks.

Network Type	Graph Theory	Optimization Theory	Game Theory	Learning Theory
Wireless		18, 19	1	
Internet	10	13, 14, 16		
Content Distribution		15, 17		
Web	3		2	4, 5
Online Social	7, 8, 9		6	
Internet Economics		11	20	12

The order of appearance of these 20 questions is arranged so that clusters of highly related topics appear next to each other. Therefore, we recommend going through the chapters in this sequence, unless you're OK with flipping back every now and then when key concepts from prior chapters are referenced. Figure 0.1 summarizes the "prerequisite" relationship among the chapters.

This book cuts across both networks among devices and networks among people. We examine networks among people that overlay on top of networks among devices, but also spend half of the book on wireless networks, content distribution networks, and the Internet itself. We'll illustrate important ideas and useful methodologies across both types of networks. We'll see striking parallels in the underlying analytic models, but also crucial differences due to domain-specific details.

We can also classify the 20 questions into three groups in terms of the stages of development in formulating and answering them:

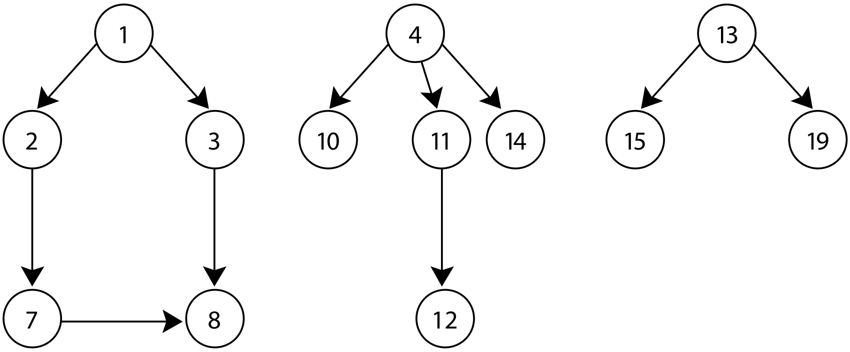


Figure 0.1 Dependence of mathematical background across some of the chapters is shown in these graphs. Each node is a chapter. Each directional link is a dependence relationship, e.g., Chapter 8's material requires that in Chapter 3 (which in turn requires that in Chapter 1) and that in Chapter 7 (which in turn requires that in Chapter 2, which in turn requires that in Chapter 1). Chapters 1, 4, and 13, the root nodes of these three trees, offer foundational material for ten other chapters. Some chapters aren't shown here because they don't form part of a dependence tree.

- Question well formulated, and theory-inspired answers adopted in practice: 1, 2, 3 4, 9, 10, 11, 13 14, 15, 16, 17, 18, 19.
- Question well formulated, but there's a gap between theory and practice (and we will discuss some possible bridges over the gaps): 12, 20.
- Question less well formulated (but certainly important to raise and explore): 5, 6, 7, 8.

It's comforting to see that most of our 20 chapters belong to the first group. Not surprisingly, questions about technological networks tend to belong to the first group, with those about social and economic networks gravitating more towards the second and third groups. It's often easier to model networked devices than networked human beings with predictive power.

Not all chapters explicitly study the impact of network topology, e.g., Chapter 7 studies influence models with decision externalities that are based on population sizes, while Chapter 8 looks at influence models with topology taken into account.

A quick word about the homework problems. There are five problems at the end of each chapter. These are a mixture of easy drills, simple extensions, challenging mini-research projects, and open-ended questions. Some important topics that we cannot readily fit into the main flow of the text are also postponed to the homework problem section. For those looking for more of the easy drills, the course website `www.network20q.com` offers additional questions.

Notation

We use **boldface** text for key terms when each is first defined. We use *italics* to highlight important, subtle, or potentially confusing points.

We use boldface math symbols to denote vectors or matrices, e.g., $\mathbf{x}$, $\mathbf{A}$. Vectors are column vectors by default. We do not use special fonts to represent sets when they are clear from the context. We use (t) to index iterations over continuous time, and $[t]$ or $[k]$ to index iterations over discrete time. We use * to denote optimal or equilibrium quantities.

Some symbols have different meanings in different chapters, because they are the standard notation in different communities.

Related Books and Courses

There's no shortage of books on networks of many kinds. The popular ones that appeared in the past decade fall into two main groups:

- Popular science books, many of them filled with historical stories, empirical evidence, and sometimes a non-mathematical sketch of technical content. Some of the widely-read ones are *Bursts, Connected, Linked, Money Lab, Planet Google, Six Degrees, Sync, The Perfect Storm, The Tipping Point*, and *The Wisdom of Crowds*. Two other books, while not exactly on networks, provide important insights into many topics in networking: *Thinking, Fast and Slow* and *The Black Swan*. On the technology networks side, there are plenty of "for dummies" books, industry certification prep books, and entrepreneurship books. There are also several history-of-technology books, e.g., *Where the Geeks Stay Up Late* and *The Qualcomm Equation.*
- Popular undergraduate- or graduate-level textbooks. On the graph-theoretic and economic side of networking, three excellent textbooks appeared in 2010: *Networks, Crowds, and Markets* by Easley and Kleinberg, *Networks* by Newman, and *Social and Economic Networks* by Jackson. The last two are more on the graduate level. An earlier popular textbook is *Social Network Analysis: Methods and Applications* by Wasserman and Faust. On the computer networking side, there's a plethora of excellent textbooks. Two particularly popular ones today are *Computer Networking: A Top-Down Approach* by Kurose and Ross, and *Computer Networks: A Systems Approach* by Peterson and Davie. On wireless communications, several textbooks published in the last few years have become popular: *Wireless Communications* by Molisch, *Wireless Communications* by Goldsmith, and *Fundamentals of Wireless Communication* by Tse and Viswanath.

As illustrated in Figure 0.2, this book fills in the gap between existing groups of books.

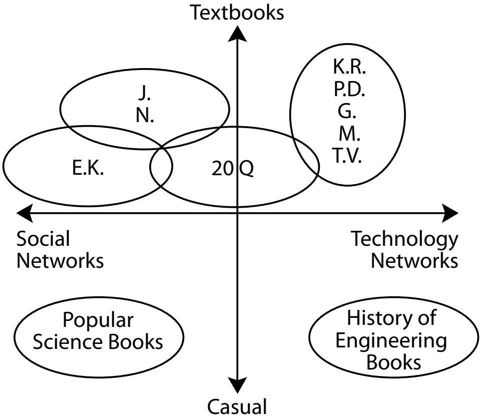


Figure 0.2 A cartoon illustrating roughly where some of the related books sit on two axes: one on the level of difficulty ranging from leisurely reading to graduate-level textbooks, and another on the mix of topics ranging from social and economic networks to technological networks. E.K. stands for Easley and Kleinberg, J. stands for Jackson, N. stands for Newman, K. R. stands for Kurose and Ross, P. D. stands for Peterson and Davie, G stands for Goldsmith, M stands for Molisch, and T.V. stands for Tse and Viswanath. 20Q stands for this book.

- Each chapter is driven by a practical question or observation, and the answers (or approximate answers) are explained using the rigorous language of mathematics. But mathematics never precedes practical problems.
- It also maintains a balance between social/economic networks and Internet/wireless networks, and between graph/economic theory and optimization/learning theory. For example, why WiFi works slower in hot spots is given as much attention as how Google auctions its ad spaces, and how IPTV networks operate is given as much detail as when information cascades initiate in a social group.
- A main goal of this book is to put social economic networks and technological networks side by side, and highlight their surprising similarities in spirit and subtle differences in detail. These examples range from the relation between Qualcomm's CDMA power control and Google's PageRank webpage ranking, to the connection between Galton's ox-weight estimation and 802.11n multiple-antenna WiFi.

These are also the differences between the Princeton undergraduate course and the seminal courses by Easley and Kleinberg at Cornell, and by Kearns at Penn. Those two courses have inspired a few similar courses at the interface between economics and computer science, such as those by by Acemoglu and Ozdaglar at MIT, by Chaintreau at Columbia, by Kempe at USC, by Parkes at Harvard, by Prabhakar at Stanford, by Spielman at Yale, by Wierman at Caltech...

These excellent courses have started structuring social and economic networking topics to undergraduates. On the other hand, both computer networking and wireless communications courses are standard, indeed often required, courses at many universities' Computer Science (CS) and Electrical Engineering (EE) departments. And across the EE, CS, Economics, Operations Research, and Applied Math departments, optimization theory, game theory, learning theory, and graph theory all have their separate courses. The new course at Princeton sits in-between the CS/Econ topics and the EE topics on networks. We hope there'll be more courses in EE and CS departments around the world that use unambigious languages to teach the concepts and methods common to social, economic, and technological networks.

Pedagogical Principles

This book and the associated course are also an experiment in three principles of teaching networks: JIT, BTP, and BAN.

Principle 1: Just In Time (JIT)

Models are often crippled by their own assumptions to start with, and frequently end up being largely irrelevant to what they set out to enable. Once in a while this is not true, but that's a low-probability event. That's why modeling is hard, especially for networks. So, before presenting any model, we first try to justify why the models are really necessary for the practical problems we face in each chapter. The material is arranged so that extensive mathematical machinery is introduced bit by bit, each bit presented just in time for the question raised. We enforce this "just-in-time" policy pretty strictly: no mathematical machinery is introduced unless it's used within the same section.

This might seem to be a rather unconventional way to write a textbook on the mathematical side of engineering. Usually a textbook asks the students to be patient with 50, 100, sometimes 200 pages of mathematics to lay the foundation first, and promises that motivating applications are coming after these pages. It's like asking a three-year-old to be patient for a long drive and promising ice-cream cones after many miles on the highway. In contrast, this book hands out an ice-cream cone every minute along the way, so that the three-year-old becomes very motivated to keep the journey going. It's more fun when gratification isn't delayed. "Fun right now" and "instant gratification" are what this book tries to achieve.

This book is an experiment motivated by this hypothesis: what professors call "fundamental knowledge" can be taught as "by-products" in the answers to practical questions that the undergraduates are interested in. A devoted sequence of lectures focusing exclusively (or predominantly) on the fundamental knowledge is not the *only* way to teach the material. Maybe we could also chop up the material and sprinkle it around. This does not "water-down" the material, it simply reorganizes it so that it shows up right next to the applications in each and every lecture. The downside is that the standard trains of thought running through the mathematical foundation of research communities are interrupted many times. This often leaves me feeling weird because I could not finish my normal teaching sequence, but the instructor feeling uncomfortable is probably a good sign. The upside is that undergraduates, who may not even be interested in a career in this field, view the course as completely driven by practical questions.

For example, the methodologies of optimization theory are introduced bit by bit in this book: linear programming and Perron-Frobenius theory in power control, convexity and least squares in Netflix recommendation, network utility maximization in Internet pricing, dynamic programming and multi-commodity flow in Internet routing, the gradient algorithm and dual decomposition in congestion control, and combinatorial optimization in peer-to-peer networks.

The methodologies of game theory are introduced bit by bit: the basic definitions in power control, auction theory in ad-space auctions, bargaining theory in Wikipedia consensus formation as well as in two-sided pricing of Internet access, and selfish maximization in tipping.

The methodologies of graph theory are introduced bit by bit: matching in ad-space auctions, consistency and PageRank in Google search, bipartite graph in Netflix recommendation, centrality, betweenness, and clustering measures in influence models, small worlds in social search, scale-free graphs in Internet topology, the Bellman–Ford algorithm and max flow min cut in Internet routing, and tree embedding in peer-to-peer networks.

The methodologies of learning theory are introduced bit by bit: collaborative filtering in Netflix recommendation, Bayesian estimation and adaptive boosting in ratings, and community detection in influence models.

Principle 2: BTP (Bridge Theory and Practice)

The size of the global industry touched upon by these 20 questions is many trillions of dollars. Just the market capitalizations of the 20 most relevant US companies to this book: Apple, Amazon, AT&T, Cisco, Comcast, Disney, eBay, EMC, Ericsson, Facebook, Google (including YouTube), Groupon, HP, Intel, LinkedIn, Microsoft (including Skype), Netflix, Qualcomm, Verizon, and Twitter added up to over $2.22 trillion as of July 4, 2012.

In theory, this book's theories are directly connected to the practice in this multi-trillion-dollar industry. In practice, that's not always true, especially in fields like networking where stable models, like the additive Gaussian noise channel for copper wire in communication theory, often do not exist.

Nonetheless, we try to strike a balance between

- presenting enough detail so that answers to these practical questions are grounded in actual practice rather than in "spherical cows" and "infinite planes," (although we couldn't help but keep "rational human beings" in several chapters), and

- avoiding too much detail that reduces the "signal-noise-ratio" in illustrating the fundamental principles.

This balance is demonstrated in the level of detail with which we treat network protocol descriptions, Wikipedia policies, Netflix recommendation algorithms, etc. And this tradeoff explains the (near) absence of random graph theory and of Internet protocols' header formats, two very popular sets of material in standard textbooks in math/CS-theory/sociology and in CS-systems/EE curricula, respectively.

Some of these 20 questions are currently trapped in particularly deep theory–practice gaps, especially those hard-to-formulate questions in Chapters 5 and 6, and those hard-to-falsify answers in Chapters 7 and 8. The network economics material in Chapters 11 and 12 also fits many of the jokes about economists, too many to quote here. (A good source of them is Taleb's *The Bed of Procrustes.*) Reverse engineering, shown across several chapters, has its own share of accurate jokes: "Normal people look at something that works in theory, and wonder if it'll also work in practice. Theoreticians look at something that works in practice, and wonder if it'll also work in (their) theory."

Time and time again, we skip the techniques of mathematical acrobats, and instead highlight the never-ending struggles between representations and realities during modeling: the process of "mathematical crystallization" where (most) parts of reality are thrown out of the window so that what remains becomes tractable using today's analytic machineries. What often remains unclear is whether the resulting answerable questions are still relevant and the resulting tractable models still have predictive powers. However, when modeling is done "right," engineering artifacts can be explained rather than just described, and better design can be carried out top-down rather than by "debug and tweak" It's often been quoted (mostly by theoreticians like me) that "there's nothing more practical than a good theory," and that "a good theory is the first-order exponent in the Taylor expansion of reality." Perhaps these can be interpreted as *definitions* of what constitutes a "good" theory. By such a definition, this book has traces of good theory, thanks to many researchers and practitioners who have been working hard on bridging the theory-practice gaps in networking.

Principle 3: BAN (Book As a Network)

Throughout the chapters, comparisons are constantly drawn with other chapters. This book itself is a network, a network of ideas living in nodes called chapters, and we grasp every opportunity to highlight each possible link between these nodes. The most interesting part of this book is perhaps this network effect among ideas: to see how curiously related, and yet crucially different they are.

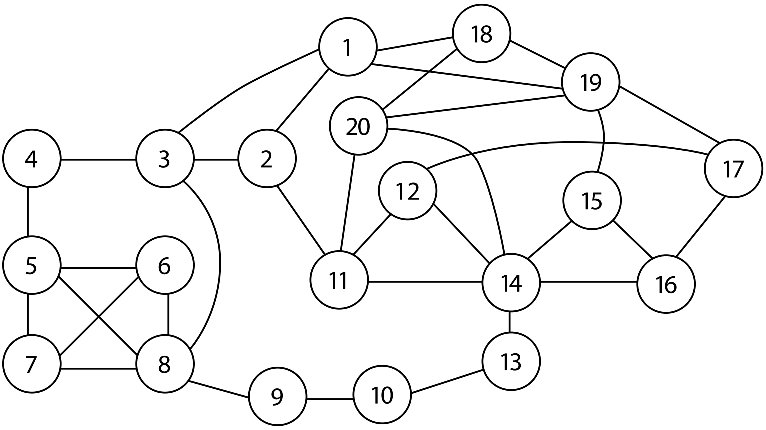


Figure 0.3 Intellectual connections across the chapters. Each node is a chapter, and each bidirectional link is an intellectual connection, via either similar concepts or common methodologies. Cliques of nodes and multiple paths from one node to another are particularly interesting to observe in this graph.

Figure 0.3 shows the main connections among the chapters. This is what the book is about: weave a network of ideas (about networks), and the positive network effect comes out of that.

We can extract the top 20 ideas across the chapters. The first 10 are features of networks, the next 5 design ideas, and the last 5 modeling approaches.

1. Resource sharing (such as statistical multiplexing and fairness): Chapters 1, 11, 13, 14, 15, 16, 17, 18, 20.
2. Opinion aggregation and consensus formation: Chapters 3, 4, 5, 6, 18.
3. Positive network effect (such as resource pooling and economy of scale): Chapters 9, 11, 13, 15, 16.
4. Negative network effect (such as tragedy of the commons): Chapters 11, 20.
5. The wisdom of crowds (diversity gain and efficiency gain): Chapters 7, 8, 18, 19.
6. The fallacy of crowds (cascade and contagion): Chapters 7, 8.
7. Functional hierarchy and layering: Chapters 13, 14, 15, 17, 19.
8. Spatial hierarchy and overlaying: Chapters 10, 13, 15, 16, 17.
9. From local actions to global property: Chapters 1, 6, 7, 8, 13, 14, 15, 18.
10. Overprovision capacity vs. overprovision connectivity: Chapters 14, 15, 16.
11. Feedback control: Chapters 1, 7, 13, 14.
12. Utility maximization: Chapters 1, 2, 11, 12, 14, 20.
13. Protocols: Chapters 14, 15, 17, 19.
14. Signaling: Chapters 6, 19.
15. Randomization: Chapters 3, 15, 18.
16. Graph consistency models: Chapters 3, 13.

17. Strategic equilibrium models: Chapters 1, 2, 15.
18. Generative model (and reverse engineering): Chapters 9, 10, 14.
19. Latent-factor models: Chapter 4.
20. Axiomatization models: Chapters 6, 20.

In the first offering of this course at Princeton, the undergrads voted (by Borda count) "resource sharing," "opinion aggregation," and "positive network effect" as the top three concepts they found most useful. They also voted the key equations in PageRank, distributed power control, and Bellman–Ford as the top three equations.

Almost every one of these 20 ideas cuts across social/economic networks and technological networks. Some examples are given below.

- The emergence of global coordination through local actions based on local views is a recurring theme, from influence models in social networks to routing and congestion control in the Internet, and from consumer reaction to pricing signals to power control in wireless networks.
- Resource sharing models, in the form of additive sharing $x + y \leq 1$, or multiplicative sharing $x/y \geq 1$, or binary sharing $x, y \in \{0, 1\}, x + y \leq 1$, are introduced for network pricing as well as the classic problems of congestion control, power control, and contention control.
- The (positive) network effect is often highlighted in social and economic networks. It also finds a concrete realization in how content is shared over the Internet through peer-to-peer protocols and how data centers are scaled up.
- "The wisdom of (independent and unbiased) crowds" is another common theme. There are two types of "wisdom" here. (1) Diversity gain in reducing the chance of some bad event (typically represented mathematically by $1 - (1 - p)^N$, where N is the size of the crowd and p the probability of some bad event). (2) Efficiency gain in smoothing out some average metric (typically represented mathematically as a factor-N in the metric). Both types are observed in social networks and in the latest generation of 4G and WiFi wireless networks.
- Consensus formation is used in computing webpage importance scores in PageRank as well as in discovering the right time to transmit in WiFi.
- Spatial hierarchy is used both in how search is done in a small world and in how the Internet is structured.
- The design methodology of feedback control is used in influence models in social networks and congestion control in the Internet.
- Utility maximization is used in auctioning advertising spots and setting Internet access pricing.
- The power method is used both in Google's PageRank and in Qualcomm's distributed power control.
- Randomization is used in PageRank and 802.11 CSMA.
- Strategic equilibrium models are used in auctioning and BitTorrent.

- Reverse engineering is used in studying scale-free networks and TCP.
- Axiomatization is used in voting procedure and fairness evaluation.

This list goes on. Yet equally important are the subtle differences between technological and socio-economic networks. Exhibit A for this alert is the (non-existence of) the Achilles' heel of the Internet and the debate between two generative models (preferential attachment vs. constrained optimization) of scale-free networks.

Two Bigger Pictures

There are two broader themes in the backdrop of this book:

- *Instill domain-specific functionalities to a generic network science.* A "network science" around these 20 questions must be based on domain-specific models and on the pursuit of falsification. For example, while a random graph is elegant, it's often neither a relevant approach to design nor the only generative model to explain what we see in this book. And as much as metrics of a static graph are important, engineering protocols governing the *functionalities* of feedback, coordination, and robustness are just as crucial as the *topological* properties of the graph like the degree distribution.
- *Revisit the Electrical and Computer Engineering (ECE) undergraduate curriculum.* In the standard curriculum in ECE since around the 1960s, a "signals and systems" course is one of the first foundational courses. As networks of various kinds play an increasingly important role both in engineering design and in the society, it's time to capture fundamental concepts in networking in a second systems course. Just as linear time-invariant systems, sampling, integral transforms, and filter design have laid the foundation of ECE curriculum since the 1960s, we think the following concepts have now become fundamental to teach to future ECE students (whether they are taught in the JIT way or not): patterns of connections among nodes, modularization and hierarchy in networked systems, consistency and consensus in graphs, distributed coordination by pricing feedback, strategic equilibrium in competition and cooperation, pros and cons of scaling up, etc.

  So this book is an experiment in both *what* to teach and *how* to teach in an ECE undergraduate course in systems: what constitutes core knowledge that needs to be taught and how to teach it in a context that enhances learning efficiency. As much as we appreciate FIR and IIR filter design, Laplace and Z transforms, etc., maybe it's about time to explore the possibility of reducing the coverage of these topics by just a tiny bit to make room for mathematical notions just as fundamental to engineering today. And we believe the best way to drive home these ideas is to tie in with applications that teenagers, and many of the older folks, use every day.

Class as a Social and Economic Network

The class "Networks: Friends, Money, and Bytes," created in parallel to this book in Fall 2011 and cross-listed in EE and CS at Princeton University, was a social and economic network itself. We tweeted, we blogged, and we created wikis. On the first day of the class, we drew a class social graph, where each node is a student, and a link represents a "know by first name before coming to the first lecture" relationship. After the last lecture, we drew the graph again.

We also created our own currency called "nuggets." The TAs and I "printed" our money as we saw fit. There were several standard ways to earn nuggets, including catching typos in lecture notes and writing popular blogs. There were ten class activities beyond homework problems that were rewarded by nuggets, including one activity in which the students were allowed to buy and sell their homework solutions using auctions. The matching of students and class project topics was also run through bidding with nuggets. Eventually the nugget balances translate into an upward adjustment of grades. To see some of the fun of ELE/COS 381 at Princeton University, visit `www.network20q.com`.

Starting in Fall 2012, this course is offered on Stanford's coursera and its own open education course website, and on YouTube and iTunes U too. The Princeton offering adopts the approach of flipped classroom advocated by the Khan Academy. With many parallel, on-going efforts from different universities and companies (CodeAcademy, coursera, EdX, udacity, etc.), it will take a few years before the landscape of open online education becomes stable. Higher education will be enhanced by this movement that has been gathering momentum since MIT's open courseware initiative in 2002. Yet many issues remain to settle at the time of writing this book: the modes and efficiency of students' learning, the boundary and reach of higher education, the prioritization and value propositions of a university's missions, the roles of faculty and the nature of classroom teaching, the differentiation between self-education and branded-certification, the authenticity and ownership of student-activity records, the tuition revenue to universities and business models of open access platforms, the drawbacks of monetization on for-profit platforms... What is already clear, however, is that this mode of education cannot be feasible without the widespread use of mobile data devices, the video-watching habit of the YouTube generation, or the crowd-sourcing of social-networked online study group. These are exactly some of the key topics studied in this course. From voting of popular questions to distributing video over 4G networks, this is a course *about* these networks and taught *through* these networks.

1 What makes CDMA work for my smartphone?

1.1 A Short Answer

Take a look at your iPhone, Android phone, or a smartphone running on some other operating system. It embodies a remarkable story of technology innovations. The rise of wireless networks, the Internet, and the web over the last five decades, coupled with advances in chip design, touchscreen material, battery packaging, software systems, business models... led to this amazing device you are holding in your hand. It symbolizes our age of networked life.

These phones have become the mobile, lightweight, smart centers of focus in our lives. They are used not just for voice calls, but also for **data applications**: texting, emailing, browsing the web, streaming videos, downloading books, uploading photos, playing games, or video-conferencing friends. The throughputs of these applications are measured in bits per second (bps). These data fly through a **cellular network** and the **Internet**. The cellular network in turn consists of the radio air-interface and the core network. We focus on the air-interface part in this chapter, and turn to the cellular core network in Chapter 19.

Terrestrial wireless communication started back in the 1940s, and cellular networks have gone through generations of evolution since the 1970s, moving into what we hear as 4G these days. Back in the 1980s, some estimated that there would be 1 million cellular users in the USA by 2000. That turned out to be one of those way-off under-estimates that did not even get close to the actual impact of networking technologies.

Over more than three decades of evolution, a fundamental concept of cellular architecture has remained essentially the same. The entire space of deployment is divided into smaller regions called **cells**, which are often represented by hexagons as in Figure 1.1, thus the name cellular networks and cell phones. There is one **base station** (BS) in each cell, connected on the one side to switches in the core network, and on the other side to the **mobile stations** (MSs) assigned to this cell. An MS could be a smart phone, a tablet, a laptop with a dongle, or any device with antennas that can transmit and receive in the right frequencies following a cellular network standard. There are a few other names for them, for example, sometimes an MS is also called a User Equipment (UE) and a BS called a Node B (NB) or an evolved Node B (eNB).

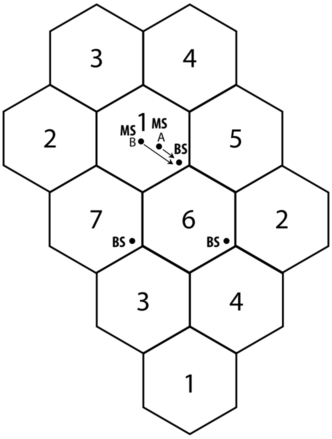


Figure 1.1 Part of a typical cellular network with a frequency reuse of 7. Each cell is a hexagon with a base station (BS) and multiple mobile stations (MSs). Only a few of them are drawn in the figure. Each BS has three directional antennas, each of which covers a 120-degree sector. Some mobile stations, like MS A, are close to the base station with strong channels to the BS. Others, like MS B, are on the cell edge with weak channels. Attenuation enables frequency reuse, but the variability of and the inability to control attenuation pose challenges to wireless cellular network design.

We see a clear hierarchy, a fixed infrastructure, and one-hop radio links in cellular networks. This is in contrast to other types of wireless networks. Moreover, the deployment of base stations is based on careful radio engineering and tightly controlled by a wireless provider, in contrast to WiFi networks in Chapter 18.

Why do we divide the space into smaller regions? Because the wireless spectrum is scarce and radio signals weaken over space.

Transmitting signals over the air means emitting energy over different parts of the electromagnetic **spectrum**. Certain regions of the spectrum are allocated by different countries to cellular communications, just like other parts of the spectrum are allocated to AM and FM radio. For example, the 900 MHz range is allocated for the most popular 2G standard called GSM, and in Europe the 1.95 GHz range and 2.15 GHz range are allocated for UMTS, a version of the 3G standard. Some part of the spectrum is unlicensed, like in WiFi, as we will see in Chapter 18. Other parts are licensed, like those for cellular networks, and a wireless service provider needs to purchase these limited resources with hefty prices. The spectrum for cellular networks is further divided into chunks, since it is often easier for transmitters and receivers to work with narrower frequency bands, e.g., on the order of 10 MHz in 3G.

The signals sent in the air become weaker as they travel over longer distances. The amount of this attenuation is often proportional to the square, or even the fourth power, of the distance traversed. So, in a typical cellular network, the signals become too weak to be accurately detected after a couple of miles. At

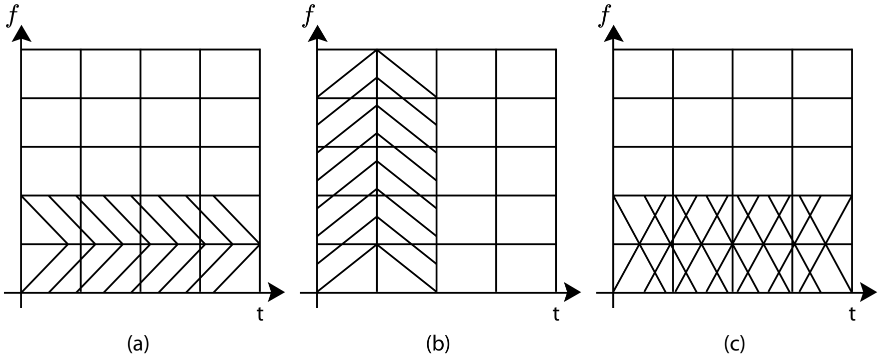


Figure 1.2 Part of a time–frequency grid is shown in each graph. For visual clarity, we only show two slices of resources being used. (a) FDMA and (b) TDMA are dedicated resource allocation: each frequency band or timeslot is given to a user. In contrast, (c) CDMA is shared resource allocation: each time–frequency bin is shared by multiple users, all transmitting and receiving over the same frequency band and at the same time. These users are differentiated by signal processing. Power control also helps differentiate their signals.

first glance, this may sound like bad news. But it also means that the frequency band used by base station A can be reused by another base station B sufficiently far away from A. All we need to do is tessellate the frequency bands, as illustrated in Figure 1.1, so that no two cells share the same frequency band if they are too close. In Figure 1.1, we say that there is a **frequency reuse factor** of 7, since we need that many frequency bands in order to avoid having two close cells sharing the same frequency band. **Cellular architecture** enables the network to scale up over space. We will visit several other ways to scale up a network later.

Now, how can the users in the *same* cell share the same frequency band? There are two main approaches: orthogonal and non-orthogonal allocation of resources.

Frequency is clearly one type of resource, and time is another. In **orthogonal allocation**, each user is given a small band of frequency in Frequency-Division Multiple Access (**FDMA**), or a timeslot in Time-Division Multiple Access (**TDMA**). Each user's allocation is distinct from others, as shown in Figure 1.2(a) and (b). This often leads to an inefficient use of resources. We will see in later chapters a recurring theme: a dedicated assignment of resources to users becomes inefficient when users come and go frequently.

The alternative, **non-orthogonal allocation**, allows all users to transmit at the same time over the same frequency band, as in Code-Division Multiple Access. **CDMA** went through many ups and downs with technology adoption from 1989 to 1995, but is now found in all the 3G cellular standards as part of the design. In CDMA's first standard, IS-95 in the 2G family, the same frequency band is reused in all the cells, as illustrated in Figure 1.2(c). But how can we distinguish the users if their signals overlap with each other?

Think of a cocktail party with many pairs of people trying to carry out individual conversations. If each pair takes turns in communicating, and only one person gets to talk during each timeslot, we have a TDMA system. If all pairs can communicate at the same time, and each uses a different language to avoid confusion, we have a CDMA system. But there are not enough languages whose pronunciations do not cause confusion, and human ears are not that good at decoding, so interference is still an issue.

How about controlling each person's volume? Each transmitter adjusts the volume of its voice according to the relative distances among the speakers and listeners. In a real cocktail party, unless there is some politeness protocol or it hurts people's vocal chord to raise their voice, we end up in a situation where everyone is shouting and yet most people cannot hear well. Transmit power control should mitigate this problem.

The core idea behind the CDMA standards follows our intuition about the cocktail party. First, the transmitter multiplies the digital signals by a sequence of 1s and minus 1s, a sequence we call the **spreading code**. The receiver multiplies the received bits by the same spreading code to recover the original signals. This is straightforward to see: 1×1 is 1, and -1×-1 is also 1. What is non-trivial is that a family of spreading codes can be designed such that only *one* spreading code, the original one used by the transmitter, can recover the signals. If you use any other spreading code in this family, you will get noise-like, meaningless bits. We call this a family of **orthogonal codes**. Users are still separated by orthogonalization, just along the "code dimension" as opposed to the more intuitive "time dimension" and "frequency dimension." This procedure is called direct sequence **spread spectrum**, one of the standard ways to enable CDMA.

However, there may not be enough orthogonal spreading codes for all the mobile stations. Families of orthogonal codes are limited in their sizes. Furthermore, a slight shift on the time axis can scramble the recovered bits at the receiver. We need the clocks on all the devices to be synchronized. But this is infeasible for the **uplink**, where mobiles talk to the base station: MSs cannot easily coordinate their clocks. It is difficult even in the **downlink**, where the base station talks to the mobiles: the BS has a single clock but the wireless channel distorts the bits. Either way, we do not have perfectly orthogonal spreading codes, even though these imperfect codes still provide significant "coding gain" in differentiating the signals.

We need an alternative mechanism to differentiate the users and to tackle the **interference** problem. Wireless signals are just energy propagating in the air, and one user's signal is every other user's interference. Interference, together with the attenuation of signals over distance and the fading of signals along multiple paths, are the the top three issues we have to address in wireless channels. Interference is an example of **negative externality** that we will encounter many times in this book, together with ways to "internalize" it by designing the right mechanism.

Here is an example of significant interference. As shown in Figure 1.1, a user standing right next to the BS can easily overwhelm another user far away at the edge of the cell. This is the classic **near-far problem** in CDMA networks. It was solved in the IS-95 standard by Qualcomm in 1989. This solution has been one of the cornerstones in realizing the potential of CDMA since then.

Qualcomm's solution to the near-far problem is simple and effective. The receiver infers the channel quality and sends that back to the transmitter as feedback. Consider an uplink transmission: multiple MSs trying to send signals to the BS in a particular cell. The BS can estimate the channel quality from each MS to itself, e.g., by looking at the ratio of the received signal power to the transmitted power, the latter being pre-configured to some value during the channel-estimation timeslot. Then, the BS inverts the channel quality and sends that value, on some feedback control channel, back to the MSs, telling them that these are the gain parameters they should use in setting their transmit powers. In this way, all the received signal strengths will be made equal. This is the basic MS **transmit power control** algorithm in CDMA.

But what if equalization of the received signal powers is *not* the right goal? For voice calls, the typical application on cell phones in 2G networks in the 1990s, there is often a target value of the received signal quality that each call needs to achieve. This signal quality factor is called the Signal to Interference Ratio (**SIR**). It is the ratio between the received signal strength and the sum strength of all the interference (plus the receiver noise strength). Of course, it is easy to raise the SIR for just one user: just increase its transmitter's power. But that translates into higher interference for everyone else, which further leads to higher transmit powers being used by them if they also want to maintain or improve their SIRs. This positive feedback escalates into a transmit power "arms race" until each user is transmitting at the maximum power. That would not be a desirable state to operate in.

If each user fixes a reasonable target SIR, can we do better than this "arms race" through a more intelligent power control? Here, "being reasonable" means that the SIRs targeted by all the users in a cell are mutually compatible; they *can* be simultaneously achieved by some configuration of transmit powers at the MSs.

The answer is yes. In 1992-1993, a sequence of research results developed the basic version of **Distributed Power Control** (DPC), a fully **distributed algorithm**. We will discuss later what we mean by "distributed" and "fully distributed." For now, it suffices to say that, in DPC, each pair of transmitter (e.g., an MS) and receiver (e.g., the BS) does not need to know the transmit power or channel quality of any other pair. At each timeslot, all it needs to know is the actual SIR it currently achieves at the receiver. Then, by taking the ratio between the fixed, target SIR and the variable, actual SIR value measured for this timeslot, and multiplying the current transmit power by that ratio, we get the transmit power for the next timeslot. This update happens simultaneously at each pair of transmitter and receiver.

This simple method is an **iterative algorithm**; the updates continue from one timeslot to the next, unlike with the one-shot, received-power-equalization algorithm. But it is still simple, and when the target SIRs can be simultaneously achieved, it has been proven to **converge**: the iterative procedure will stop over time. When it stops, it stops at the right solution: a power-minimal configuration of transmit powers that achieves the target SIRs for all. DPC converges quite fast, approaching the right power levels with an error that decays as a geometric series. DPC can even be carried out asynchronously: each radio has a different clock and therefore different definitions of what timeslot it is now.

Of course, in real systems the timeslots are indeed *asynchronous* and power levels are *discrete*. Asynchronous and quantized versions of DPC have been implemented in all the CDMA standards in 3G networks. Some standards run power control 1500 times every second, while others run 800 times a second. Some discretize power levels to 0.1 dB, while others between 0.2 and 0.5 dB. Without CDMA, our cellular networks today could not work as efficiently. Without power control algorithms (and the associated handoff method to support user mobility), CDMA could not function properly. In Chapter 19, we will discuss a 4G standard called LTE. It uses a technology called OFDM instead of CDMA, but power control is still employed for interference reduction and for energy management.

Later, in Chapter 18, we will discuss some of the latest ideas that help further push the data rates in new wireless network standards, ranging from splitting, shrinking, and adjusting the cells to overlaying small cells on top of large ones for traffic offloading, and from leveraging multiple antennas and tilting their positions to "chopping up" the frequency bands for more efficient signal processing.

1.2 A Long Answer

1.2.1 Distributed power control

Before we proceed to a general discussion of the Distributed Power Control (DPC) algorithm, we must first define some symbols.

Consider N pairs of transmitters and receivers. Each pair forms a (logical) link, indexed by i. The transmit power of the transmitter of link i is p_i, some positive number, usually capped at a maximum value: $p_i \leq p_{max}$ (although we will not consider the effect of this cap in the analysis of the algorithm). The transmitted power impacts both the received power at the intended receiver and the received interference at the receivers of all other pairs.

Now, consider the channel from the transmitter of link (i.e., transmitter–receiver pair) j to the receiver of link i, and denote the **channel gain** by G_{ij}. So G_{ii} is the direct channel gain; the bigger the better, since it is the channel for the intended transmission for the transmitter–receiver pair of link i. All the other $\{G_{ij}\}$, for j not equal to i, are gains for interference channels, so the smaller the better. We call these channel "gains", but actually they are less than 1, so maybe a better term is channel "loss."

This notation is visualized in Figure 1.3 for a simple case of two MSs talking to a BS, which can be thought of as two different (logically separated) receivers physically located together.

2 How does Google sell ad spaces?

2.1 A Short Answer

Much of the web services and online information is “free” today because of the advertisements shown on the websites. It is estimated that the online ad industry worldwide reached $94.2 billion in 2012. Compared with traditional media, online advertisements’ revenue ranked right below TV and above newspapers.

In the early days of the web, i.e., 1994-1995, online advertisements were sold as banners on a per-thousand-impression basis. But seeing an ad does not mean clicking on it or buying the advertised product or service afterwards. In 1997, GoTo (which later became Overture) started selling advertisement spaces on a per-click basis. This middle ground between ad revenue (what the website cares about) and effectiveness of ad (what the advertisers care about) became a commonly accepted foundation for online advertising.

With the rise of Google came one of the most stable online ad market segments: **search ads**, also called *sponsored search*. In 2002, Google started the AdWords service where you can create your ad, attach keywords to it, and send it to Google’s database. When someone searches for a keyword, Google will return a list of search results, as well as a list of ads on the right panel, or even the main panel, if that keyword matches any of the keywords of ads in its database. This process takes place continuously and each advertiser can adjust her bids frequently. There are often many ad auctions happening at the same time too. We will skip these important factors in the basic models in this chapter, focusing just on a single auction.

Now we face three key questions. First, where will your ad appear on the list? We all know that the order of appearance makes a big difference. You will have to pay more to have your ad placed higher in the list. For example, when I did a search for “Banff National Park” on Google in September 2011, I saw an ad for `www.banfflakelouise.com`, a vacation-planning company. This ad was right on top of the main panel, above all the search results on websites and images of Banff National Park. (By the way, how those “real” search results are ordered is the subject of the next chapter.) You also see a list of ads on the right panel, starting with the top one for `www.rockymountaineer.com`, a tourist train company. These two companies probably get most of the clicks, and pay more than the other advertisers for each click. The rest of this chapter delves into the auction methods that allocate these ad spaces according to how much each advertiser is willing to pay.

Second, when will these advertisers pay Google? Only when someone clicks on the link and visits their websites (like I just did, thus contributing to Google's revenue). The average number of times that a viewer of the search result page clicks an ad link, over say one hour, is called the **clickthrough rate**. In a general webpage layout, it may be difficult to rank ad spaces by their positions along a line, but we can always rank them by their clickthrough rates. Let us say the payment by advertisers to Google is *proportional* to the clickthrough rates.

Third, what is in it for the advertisers then? Their revenue derived from placing this particular ad is the product of two things: C, the number of clicks (per unit time, say, one hour), and R, the average revenue (in dollars) generated from each click.

- Let us assume that the number of clicks per hour actually observed is indeed the estimated clickthrough rate, both denoted as C. This is of course not true in general, but it is a reasonable assumption to make for our purpose. We also assume that C is independent of the content in the actual advertisement placed, again a shaky assumption to make the model more tractable.
- As for the average revenue R generated from each click (averaged over all clicks), that highly depends on the nature of the goods or services being advertised and sold. R for each ad-space buyer is assumed to be independent of which ad space she ends up buying, a more reasonable assumption than the one on the independence of C of the advertisement content.

This product, $C \times R$, is the buyer's expected revenue from a particular ad space. It is the **valuation** of the ad space to the buyer. For example, if C is 20 clicks per hour for an ad space and R is \$8 generated per click for an ad-space buyer, the valuation of that space to this buyer is \$160 (per hour). For multiple ad spaces, the valuation of each buyer is a *vector*, with one entry per ad space.

In this discussion, there is one seller, which is Google, many buyers/bidders (the advertisers), and many "goods" (the ad spaces). Each bidder can bid for the ad spaces, and Google will then allocate the ad spaces among the bidders according to some rule, and charge the bidders accordingly.

This process is an **auction**. In general, you can have S sellers, N bidders, and K items in an auction. We will consider only the case with $S = 1$. An ad space auction is a special case of general auctions. Auctions can be analyzed as games, i.e., with a set of players, a strategy set per player, and a payoff function per player.

Depending on the rules of an auction, each bidder may choose to bid in different ways, maybe bidding her true valuation of the ad spaces. It would be nice to design the rules so that such a **truthful bidding** behavior is encouraged. But Google has other considerations too, such as maximizing its total revenue: the sum of the revenue from each bidder, which is in turn the product of the number

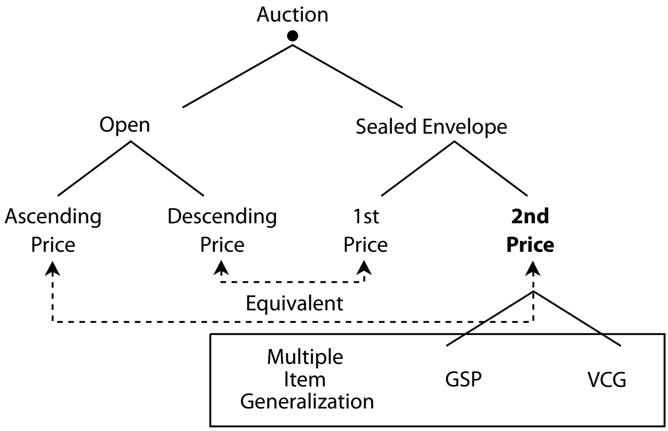


Figure 2.1 A taxonomy of major types of auction in this chapter. The second price, sealed envelope auction is equivalent to (a simpler version of) ascending price open auction, and can be generalized in two different ways to multiple-item auctions: (1) a simple extension to the Generalized Second Price (GSP) auction, and (2) a more sophisticated extension to the Vickrey–Clarke–Groves (VCG) auction that preserves the truthful bidding property.

of clicks (actually observed in real time) and the per-click charge (determined during the auction).

Before we analyze a K-item auction, we will first study auctions with only $K = 1$ item. There are two main types of such one-item auctions: ascending-price and descending-price. These intuitive types of auction, in use since the Roman Empire, require a public venue for announcing the bids.

- In an **ascending price auction**, an auctioneer announces a base price, and then each bidder can raise her hand to bid a higher price. This price war keeps escalating until one bidder submits a price, no other bidder raises a hand, and the auctioneer calls out "gone." The last bidder is the winning bidder, and she pays the price she bid in the last round.
- In a **descending price auction**, an auctioneer announces a high price first, so high that no bidder is willing to accept it. The auctioneer then starts to lower the price, until there is one bidder who shouts out "OK." That bidder is allocated the item, and pays the price announced when she said "OK."

The alternative to a public venue is private disclosure of bids, called **sealed envelope** auctions. This is much more practical in many settings, including selling ad spaces by Google and auctioning goods on eBay. There are two types of such auctions, but it turns out that their results are essentially equivalent to the two types of open auctions we just discussed.

Each bid b_i is submitted by bidder i in a sealed envelope. All bids are then revealed simultaneously to the auctioneer, who will then decide

- the allocation, and
- how much to charge for each item.

The allocation part is easy: the highest bidder gets the item; but the amount charged can vary.

- In a **first price auction**, the winner pays the highest bid, i.e., her own bid.
- In a **second price auction**, the winner pays the second highest bid, i.e., the bid from the next highest bidder.

Second price auction sounds “wrong.” If I know I will be paying the next highest bid, why not bid extremely high so that I can win the item, and then pay a much lower price for it? As it turns out, this intuition *itself* is wrong. The assumption of “much lower prices being bid by other bidders” does not hold when everyone engages in the same strategic thinking.

Instead, a second price auction is effectively equivalent to the highly intuitive ascending price auction, and can induce truthful-bidding behavior from the bidders. That is why second price auction is used so often, from auctioning major municipal projects to auctioning wireless spectrum.

Finally, we come back to auctions of K items (still with 1 seller and N bidders). If we follow the basic mechanism of second price auction, we obtain what is called the **Generalized Second Price** (GSP) for ad space auction: the ith ad space goes to the bidder that puts in the ith highest bid, and the charge, per clickthrough rate, is the $(i+1)$th bid. If the webpage layout shows the ads vertically, the advertiser in a given ad space is paying a price that is the same as the bid from the advertiser in the ad space *right below* hers. This simple method is used by Google in selling its ad spaces.

But it turns out GSP is *not* an auction that induces truthful bidding, and there can be many Nash equilibria if we analyze it as a game. An alternative is the **Vickrey–Clarke–Groves** (VCG) auction, which actually extends the second price auction’s property of truthful bidding to multiple-item auctions. A VCG auction charges on the basis of negative externality, a principle that we will see many times throughout this book. The relationships between these types of auction are summarized in Figure 2.1.

Throughout the chapter, we focus on the simplest case, where there is a single round of bidding. In reality, there are multiple related bids going on at the same time, e.g., `www.banfflakelouise.com` may be bidding for multiple related keywords, such as “Banff,” “Lake Louise,” and “Canadian vacation,” simultaneously. In a homework problem, we will go into a little more detail on one aspect of simultaneous auction in the context of spectrum auctioning.

3 How does Google rank webpages?

3.1 A Short Answer

Now we turn to the other links you see on a search-result webpage; not the ads or sponsored search results, but the actual ranking of webpages by search engines such as Google. We will see that, each time you search on `www.google.com`, Google solves a very big system of linear equation to rank the webpages.

The idea of embedding links in text dates back to the middle of the last century. As the Internet scaled up, and with the introduction of the web in 1989, the browser in 1990, and the web portal in 1994, this vision was realized on an unprecedented scale. The network of webpages is huge: somewhere between 40 billion and 60 billion according to various estimates. And most of them are connected to each other in a giant component of this network. It is also sparse: most webpages have only a few hyperlinks pointing inward from other webpages or pointing outward to other webpages. Google search organizes this huge and sparse network by ranking the webpages.

More important webpages should be ranked higher. But how do you quantify *how* important a webpage is? Well, if there are many other important webpages pointing towards webpage A, A is probably important. This argument implicitly assumes two ideas:

- Webpages form a network, where a webpage is a node, and a hyperlink is a *directed* link in the network: webpage A may point to webpage B without B pointing back to A.
- We can turn the seemingly circular logic of "important webpages pointing to you means you are important" into a set of equations that characterize the *equilibrium* (a fixed-point equilibrium, not a game-theoretic Nash equilibrium) in terms of a *recursive definition* of "importance." This importance score will then act as an approximation of the ultimate test of search engines: how useful a user finds the search results.

As mentioned in Chapter 1, a network consists of both a *topology* and *functionalities.* Topology is often represented by a graph and various matrices, several of which will be introduced in this chapter and a few more in later chapters. And we will assume some models of the "search and navigation" functionality in this chapter.

Suppose there are N webpages. Each webpage i has O_i **outgoing links** and I_i **incoming links**. We cannot just count the number of webpages pointing to a given webpage A, because that number, the **in-degree** of the node in the hyperlinked graph, is often not the right measure of importance.

Let us denote the "importance score" of each webpage by π_i. If important webpages point to webpage A, maybe webpage A should be important too, i.e., $\pi_A = \sum_{i \to A} \pi_i$, where the sum is taken over all the webpages pointing to A. However, this is not quite right either, since node i may be pointing to many other nodes in this graph, and that means each of these nodes receives only a small portion of node i's importance score.

Let us assume that each node's importance score is *evenly* distributed across all the outgoing links from that node, i.e., each of the outgoing neighbors of node i receives π_i/O_i importance score. Now each node's importance score can also be written as the sum of the importance scores received *from* all of the incoming neighbors, indexed by j, e.g., for node 1,

$$\sum_{j \to 1} \frac{\pi_j}{O_j}.$$

If this sum is indeed also π_1, we have *consistency* of the scores. But it is not clear whether we can readily compute these scores, or even whether there is a consistent set of scores at all.

It turns out that, with a couple of modifications to the basic idea above, there is always a unique set of consistent scores, denoted as $\{\pi_i^*\}$. These scores determine the ranking of the webpages: the higher the score, the higher the webpage is ranked.

For example, consider a very small graph with just four webpages and six hyperlinks, shown in Figure 3.1. This is a directed graph where each node is a webpage and each link a hyperlink. A consistent set of importance scores turns out to be (0.125, 0.125, 0.375, 0.375): webpages 3 and 4 are more important than webpages 1 and 2. In this small example, it so happens that webpages 3 and 4, linking each other, push both webpages' rankings higher.

Intuitively, the scores make sense. First, by symmetry of the graph, webpages 1 and 2 should have the same importance score. We can view webpages 3 and 4 as if they form one webpage first, a supernode 3+4. Since node 3+4 has two incoming links, and each of nodes 1 and 2 has only one incoming link, node 3+4 should have a higher importance score. Since node 3 points to node 4 and vice versa, these two nodes' importance scores mix into an equal division at equilibrium. This line of reasoning qualitatively explains the actual scores we see.

But how do we calculate the exact scores? In this small example, it boils down to two simple linear equations. Let the score of node 1 (and 2) be x, and that of node 3 (and 4) be y. Looking at node 1's incoming links, we see that there is only one such link, coming from node 4 that points to three nodes. So we know

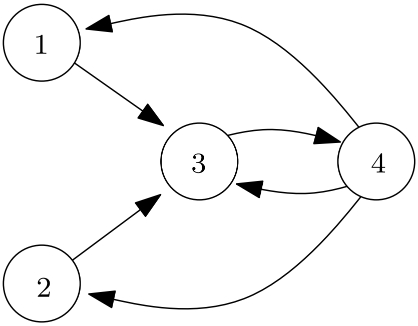


Figure 3.1 A simple example of importance score with four webpages and six hyperlinks. It is a small graph with much symmetry, leading to a simple calculation of the importance scores of the nodes.

$x = y/3$. By normalization, all scores must add up to $2x + 2y = 1$. So we have $x = 0.125$ and $y = 0.375$.

Now, how do we compute this set of consistent scores in a large, sparse, general graph of hyperlink connectivity?

3.2 A Long Answer

In any search engine, there are two main activities constantly occurring behind the scenes: (1) crawling the hyperlinked web space to get the webpage information, and (2) indexing this information into concise representations and storing the indices.

When you search in Google, it triggers a ranking procedure that takes into account two main factors:

- a **relevance score**: how relevant to the search the content is on each webpage, and
- an **importance score**: how important the webpage is.

It is the composite score of these two factors that determines the ranking. We focus on the importance score, since that usually determines the order of the top few webpages in any reasonably popular search, and has a tremendous impact on how people obtain information and how online businesses generate traffic.

We will be constructing several related matrices: $\mathbf{H}, \hat{\mathbf{H}}$, and $\mathbf{G}$, step by step (this matrix $\mathbf{G}$ is not the channel gain matrix of Chapter 1; it denotes the Google matrix in this chapter). Eventually, we will be computing an eigenvector of $\mathbf{G}$ as the importance-score vector. Each matrix is $N \times N$, where N is the number of the relevant webpages. These are extremely large matrices, and we will discuss the computational challenge of scaling-up in the Advanced Material.

3.2.1 Constructing $\mathbf{H}$

The first matrix we define is $\mathbf{H}$: its (i, j)th entry is $1/O_i$ if there is a hyperlink from webpage i to webpage j, and 0 otherwise. This matrix describes the network

4 How does Netflix recommend movies?

We just saw three beautiful equations in the last three chapters, each used at least a billion times every single day:

$$
\begin{aligned}
p_i[t+1] &= \frac{\gamma_i}{\mathrm{SIR}_i[t]} p_i[t], \\
U_i(\mathbf{b}) &= v_i - p_i(\mathbf{b}), \\
\pi^{*T} &= \pi^{*T}\mathbf{G}.
\end{aligned}
$$

We continue with our first block of four chapters that present four fundamental algorithms: distributed power control, second price auction, PageRank, and now, collaborative filtering. These four chapters also introduce the basic languages of optimization, game, graph, and learning theories. A word of caution: as a chapter that introduces the basic ideas both in convex optimization and in machine learning, this chapter is among the longest in the book; you have to wait about 14 pages before we get to the most important idea on collaborative filtering for Netflix. This chapter is also mathematically more demanding than most others.

4.1 A Short Answer

4.1.1 Recommendation problem

Netflix started its DVD rental business in 1997: instead of going to rental stores, you can just wait for DVDs to arrive by mail. Instead of incurring a late fee for each day you hold the DVD beyond the return date, you can keep holding the DVD as long as you continue to pay the monthly subscription fee, but you cannot receive a new DVD without returning the old one. This is similar in spirit to the sliding window mechanism of congestion control in Chapter 14, or the tit-for-tat incentive mechanism of P2P in Chapter 15. Netflix also maintained an efficient inventory control and mail delivery system. It operated with great *scalability* (i.e., the per-customer cost is much lower as the number of customers goes up) and *stickiness* (i.e., users are reluctant to change the service). By 2008, there were about 9 million users in the USA and Canada.

Then Netflix moved on to the next mode of delivering entertainment. This time it was streaming movies and TV programs from video servers, through the Internet and wireless networks, to your Internet-connected devices: TVs, set-top boxes, games consoles, smartphones, and tablets. With its branding, choice of content, and aggressive pricing, Netflix's subscriber base nearly tripled to 23 million by April 2011. Netflix video streaming generated so much Internet traffic that over one in every four bits going through the Internet that month was Netflix traffic. In September 2011, Netflix announced that it would separate the DVD rental and online streaming businesses. Soon afterwards, Netflix reversed the decision, although the pricing for DVD rental and for online streaming became separated.

We will look at cloud-based video-distribution services, including Netflix, Amazon Prime, Hulu, HBO Go, etc., in Chapter 17, and how that is changing the future of both entertainment and networking. In this chapter, we instead focus on the social-network dimension used by Netflix: How does it recommend movies for you to watch (either by mail or by streaming)? It is like trying to read your mind and predict your movie rating. An effective **recommendation system** is important to Netflix because it enhances user experience, increases loyalty and volume, and helps with inventory control.

A recommendation system is a helpful feature for many applications beyond video distribution. Just like search engines in Chapter 3, recommendation systems give rise to structures in a "sea" of raw data and reduce the impact of information "explosion." Here are some representative systems of recommendation.

- You must have noticed how Amazon recommends products to you on the basis of your purchase and viewing history, adjusting its recommendation each time you browse a product. Amazon recommendation runs **content-based filtering**, in contrast to the **collaborative filtering** used by Netflix. (A related question is when can you trust the averaged rating of a product on Amazon? This is a different variant of the recommendation problem, and will be taken up in Chapter 5.)
- You must have also been swayed by recommendations on YouTube that followed each of the videos you watched. We will look at YouTube recommendation in Chapter 7.
- You may have used Pandora's online music selection, where recommendation is developed by experts of music selection. But you get to thumbs-up or thumbs-down the recommendation in the form of an explicit, binary feedback.

Netflix instead wants to develop a recommendation system that does *not* depend on any expert, but uses the rich history of *all* the user behaviors to profile *each* user's taste in movies. This system has the following inputs, outputs, and criteria of success.

- Among the inputs to this system is the history of star ratings across all the users and all the movies. Each data point consists of four numbers: (1) user ID, indexed by u, (2) movie title, indexed by i, (3) number of stars, $1-5$, in the rating, denoted as r_{ui}, and (4) date of the rating, denoted as t_{ui}. This is a really large data set: think of millions of users and tens of thousands of movies. But only a fraction of users will have watched a given movie, and only a fraction of that fraction actually bothered to rate the movie. Still, the size of this input is on the order of billions for Netflix. And the data set is also biased: knowing which users have watched and rated which movies already provides much information about people's movie taste. For a less popular service, there would also have been a cold-start problem: too little data to start with.

- The output is, first of all, a set of predictions $\hat{r}_{ui}$, one for each movie i that user u has not watched yet. These can be real numbers, not just integers like an actual rating r_{ui}. We can interpret a predicted rating of, say, 4.2 as saying that the user will rate this movie 4 stars with 80% probability and 5 stars with 20% probability. The final output is a short, rank-ordered list of movies recommended to each user u, presumably those movies receiving $\hat{r}_{ui} \geq 4$, or the top five movies with the highest predicted $\hat{r}_{ui}$.
- The real test of this mind-reading system is whether user u actually likes the recommended movies. This information, however, is hard to collect. So a proxy used by Netflix is the **Root Mean Squared Error** (RMSE), measured for those (u, i) pairs for which we have both the prediction and the actual rating. Let us say there are C such pairs. Each rating prediction's error is squared: $(r_{ui} - \hat{r}_{ui})^2$, and then averaged over all the predictions. Since the square was taken, to scale the numerical value back down, a square root is taken over this average:

$$\text{RMSE} = \sqrt{\sum_{(u,i)} \frac{(r_{ui} - \hat{r}_{ui})^2}{C}}.$$

The smaller the RMSE, the better the recommendation system. Netflix could have used the absolute value of the error instead of the squared error, but for our purposes we will stick to RMSE as the metric that quantifies the accuracy of a recommendation system. More importantly, regardless of the error metric it uses, in the end only the ranked order of the movies matters. Only the top few in that rank-ordered list are relevant as only they will be recommended to the user. The ultimate test is whether the user decides to watch the recommended movies, and whether she likes them or not. So, RMSE minimization is just a tractable approximation of the real problem of recommendation.

4.1.2 The Netflix Prize

Could recommendation accuracy be improved by 10% over what Netflix was using? That was the question Netflix presented to the research community in

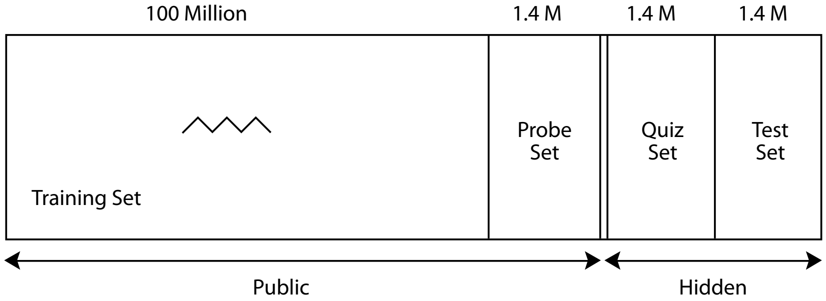


Figure 4.1 The Netflix Prize's four data sets. The training set and probe set were publicly released, whereas the quiz set and test set were hidden from the public and known only to Netflix. The probe, quiz, and test sets had similar statistical properties, but the probe set could be used by each competing team as often as they want, and the quiz set at most once a day. The final decision was based on comparison of the RMSE on the test set.

October 2006, through an open, online, international competition with a $1 million prize called the **Netflix Prize**.

The competition's mechanism is interesting in its own right. Netflix made available a set of over 100 million ratings, as part of its records from 1999 to 2005. That amount of data could fit in the memory of standard desktops in 2006, making it easy for anyone in the world to participate in the competition. The rating data came from more than 480000 users and 17770 movies. On average, each movie was rated by more than 5000 users and each user rated more than 200 movies. But those average numbers disguise the real difficulty here: many users rated only a few movies, and very few users rated a huge number of movies (one user rated over 17000 movies). Whatever recommendation system we use, it must work well for *all* users.

The exact distribution of the data is shown in Figure 4.1.

- A little fewer than 100 million ratings were made public as the *training set*.
- About 1.4 million additional ratings were also made public and they had similar statistical properties to the test set and the quiz set described next. This set of ratings was called the *probe set*, which competitors for the Netflix Prize could use to test their algorithms.
- About 1.4 million additional ratings were hidden from the competitors; this set was called the *quiz set*. Each competing team could submit an algorithm that would run on the quiz test, but not more than once a day. The RMSE scores continuously updated on the leaderboard of the Netflix Prize's website were based on this set's data.
- Another 1.4 million ratings, also hidden from the competitors, were called the *test set*. This was the real test. The RMSE scores on this set would determine the winner.

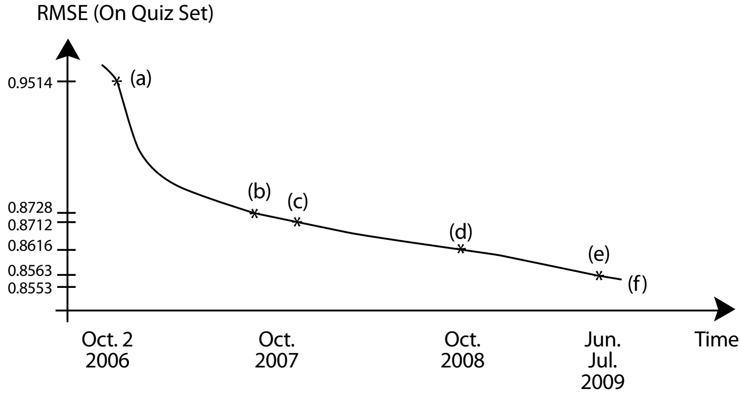


Figure 4.2 The Netflix Prize's timeline and some of the highlight events. It lasted for almost three years and the final decision came down to a 20-minute differential. The y-axis shows the progress towards reducing the RMSE on the quiz set data by 10% relative to the benchmark.

Each competing team first came up with a model for its recommendation system. Then it decided its model parameters' values by minimizing the RMSE between the known ratings in the training set and their model's predictions. Finally, it used this model with tuned parameters to predict the unknown ratings in the quiz set. Of course, Netflix knew the actual ratings in the quiz set, and could evaluate the RMSE between those ratings and the predictions from each team.

This was a smart arrangement. No team could reverse engineer the actual test set, since only scores on the quiz set were shown. It was also helpful to have a probe set on which the competing teams could run their own tests as many times as they wanted.

Netflix had its own algorithm called Cinematch that gave an RMSE of 0.9514 on the quiz set if its parameters were tuned by the training set. Improving the RMSE by even 0.01 could sometimes make a difference in the top ten recommendations for a user. If the recommendation accuracy could be improved by 10% over Cinematch, it would push RMSE to 0.8563 on the quiz set, and 0.8572 on the test set.

This Netflix Prize ignited the most intense and high-profile surge of activities in the research communities of machine learning, data mining, and information retrieval in recent years. To some researchers, the quality and sheer amount of the available data were as attractive as the hype and prize. Over 5000 teams worldwide entered more than 44000 submissions. Both Netflix and these research fields benefited from the three-year quest towards the goal of 10%. It turned out that setting the target as a 10% improvement was a really good decision. For the given training set and quiz set, getting an 8% improvement was reasonably easy, but getting a 11% would have been extremely difficult.

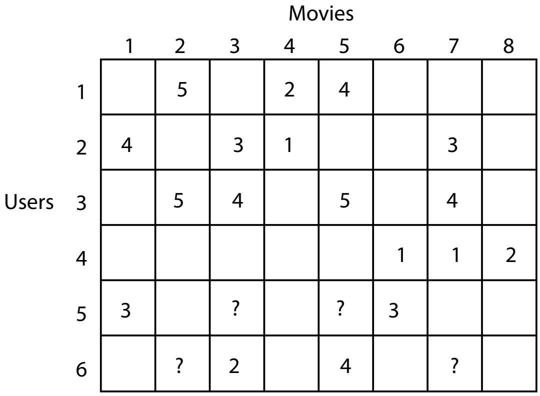
Here are a few highlights in the history of the Netflix Prize, also shown in the timeline in Figure 4.2.

- (a) Within a week of the start of the competition, Cinematch was beaten.
- (b) By early September, 2007, team BellKor made an 8.26% improvement over Cinematch, but the first place changed hands a couple of times, until
- (c) in the last hour before the first year of competition ended, the same team got 8.43% improvement and won the $50,000 annual progress prize for leading the pack during the first year of the competition.
- Then teams started merging. BellKor and BigChaos, two of the leading teams, merged and (d) received the 2008 progress prize for pushing the RMSE down to 0.8616. They further merged with Pragmatic Theory, and
- (e) in June 2009, the new team, BellKor's Pragmatic Chaos, became the first team to achieve more than 10% improvement, beating Cinematch by 10.06%, on the quiz set.
- Then the competition entered the "last call" period: all teams had 30 days to make their final submissions. (f) At the end of this period, two teams beat Cinematch by more than 10% on the quiz set: BellKor's Pragmatic Chaos had an RMSE of 0.8554, and The Ensemble had an RMSE of 0.8553, slightly better. The final winner was to be declared by comparing their RMSEs on the test set.
- Here is the grand finale: both teams beat Cinematch by more than 10% on the test set, and actually got the same RMSE on that set: 0.8567. But BellKor's Pragmatic Chaos submitted their algorithm 20 minutes earlier, and thus became the winner of the grand prize. A world-class science race lasting almost three years concluded with a 20-minute differential.

You must be wondering what algorithm BellKor's Pragmatic Chaos used in the final winning solution. The answer is documented in detail in a set of three reports, one from each component of this composite team, linked from the Netflix Prize website. But what you will find is that the winning solution was really a cocktail of many methods combined, with hundreds of ingredient algorithms blended together and thousands of model parameters fine-tuned specifically to the training set provided by Netflix. That was what it took to get that last 1% of improvement. But if you are interested only in the main approaches, big ideas, and getting $8-9\%$ improvement over Cinematch, there are actually just a few key methodologies. Those are what we will focus on in the rest of this chapter.

4.1.3 Key ideas

To start with, take a look at the table in Figure 4.3. We can also think of the table as a matrix $\mathbf{R}$, or as a *weighted* bipartite graph where the user nodes are on the left column and the movie nodes on the right. There is a link connecting user node u and movie node i if u rated i, and the value of the rating is the



		Movies							
		1	2	3	4	5	6	7	8
	1		5		2	4			
	2	4		3	1			3	
Users	3		5	4		5		4	
	4						1	1	2
	5	3		?		?	3		
	6		?	2		4		?	

Figure 4.3 Recommendation system's problem: predicting missing ratings from given ratings in a large yet sparse table. In this small example of six users and eight movies, there are eighteen known ratings as a training set, and four unknown ratings to be predicted. Real problems are much larger (with billions of cells in the table) and much sparser (only about 1% filled with known ratings).

weight of the link. In Chapter 8 we will discuss other matrices that describe the structure of different graphs.

Each column in this table is a movie (or an item in general), each row is a user, and each cell's number is the star rating by that user for that movie. Most cells are empty, since only a few users rated a given movie. You are given a large yet sparse table like this, and asked to predict some missing entries like those four indicated by question marks in the last two rows in this table.

There are two main types of techniques for any recommendation system: content-based filtering and collaborative filtering.

Content-based filtering looks at each row in isolation and attaches labels to the columns: if you like a comedy with Rowan Atkinson, you will probably like another comedy with Rowan Atkinson. This straightforward solution is often inadequate for Netflix.

In contrast, collaborative filtering exploits all the data in the entire table, across all the rows and all the columns, trying to discover *structures* in the pattern across the table. Drawing an imperfect analogy with search engines in Chapter 3, content-based filtering is like the relevance score of individual webpages, and collaborative filtering is like the importance score determined by the connections among the webpages.

In collaborative filtering, there are in turn two main approaches.

- The intuitively simpler one is the **neighborhood model**. Here, two users are "neighbors" if they share similar tastes in movies. If Alice and Bob both like "Schindler's List" and "Life is Beautiful," but not as much "E.T." and "Lion King," then knowing that Alice likes "Dr. Zhivago" would make us think Bob likes "Dr. Zhivago," too. In the neighborhood method, we first compute a similarity score between each pair of users. The larger the score, the closer these two users are in their taste for movies. Then for a given user whose opinion of a movie we would like to predict, we select, say, 50 of the most similar users who have rated that movie. Then take a weighted sum of these ratings and call that our prediction.

- The second approach is called the **latent-factor model**. It assumes that underneath the billions of ratings out there, there are only a few hundred key factors on which users and movies interact. Statistical similarities among users (or among movies) are actually due to some hidden, low-dimensional structure in the data. This is a big assumption: that there are many fewer *types* of people and movies than there are people and movies, but it sounds about right. It turns out that one way to represent a low-dimensional model is to factorize the table into two sets of *short vectors* of "latent factors."

Determining baseline predictors for the neighborhood model, or finding just the right short vectors in the latent-factor model, boils down to solving **least squares** problems, also called **linear regressions**.

Most of the mileage in the leading solutions to the Netflix Prize was obtained by combining variants of these two approaches, supplemented by a whole bag of tricks. Two of these supplementary ideas are particularly interesting.

One is **implicit feedback**. A user does not have to rate a movie to tell us something about her mind. Which movies she browsed, which ones she watched, and which ones she bothered to rate at all are all helpful hints. For example, it is useful to leverage the information in a binary table where each entry simply indicates whether this user rated that movie or not.

Another idea played an important role in pushing the improvement to 9% in the Netflix Prize: incorporating *temporal dynamics*. Here, the model parameters become time-dependent. This allows the model to capture changes in a person's taste and in trends of the movie market, as well as the mood of the day when a user rated movies, at the expense of dramatically increasing the number of model parameters to optimize. One interesting observation is that when a user rates many movies on the same day, she tends to give similar ratings to all of these movies. By discovering and discounting these temporal features, the truly long-term structures in the training set are better quantified.

In the next section, we will present baseline predictor training and the neighborhood method, leaving the latent-factor model to the Advanced Material.

4.2 A Long Answer

Before diving into specific predictors, let us take a look at the generic workflow consisting of two phases, as shown in Figure 4.4.

- *Training*: We put in a model (a mathematical representation of what we want to understand) with its parameters to work on the observed data, and

5 When can I trust an average rating on Amazon?

In this and the next three chapters, we will walk through a remarkable landscape of intellectual foundations. But sometimes we will also see significant gaps between theory and practice.

5.1 A Short Answer

We continue with the theme of recommendation. Webpage ranking in Chapter 3 turns a graph into a rank-ordered list of nodes. Movie ranking in Chapter 4 turns a weighted bipartite user–movie graph into a set of rank-ordered lists of movies, with one list per user. We now examine the aggregation of a vector of rating scores by reviewers of a product or service, and turn that vector into a scalar for each product. These scalars may in turn be used to rank order a set of similar products. In Chapter 6, we will further study aggregation of many vectors into a single vector.

When you shop on Amazon, likely you will pay attention to the number of stars shown below each product. But you should also care about the number of reviews behind that averaged number of stars. Intuitively, you know that a product with two reviews, both 5 stars, might not be better than a competing product with one hundred reviews and an average of 4.5 stars, especially if these one hundred reviews are all 4 and 5 stars and the reviewers are somewhat trustworthy. We will see how such intuition can be sharpened.

In most online review systems, each review consists of three fields:

1. rating, a numerical score often on the scale of 1–5 stars (this is the focus of our study),
2. review, in the form of text, and
3. review of review, often a binary up or down vote.

Rarely do people have time to read through all the reviews, so a summary review is needed to aggregate the individual reviews. In particular, we need to aggregate a vector of rating numbers into a single number, so that a ranking of similar products can be generated from these ratings. What is a proper aggregation? That is the subject of this chapter.

Reviews are sometimes not very trustworthy, yet they are important in so many contexts, from peer reviews in academia to online purchases of every kind. The hope is that the following two approaches can help.

First, we need methods to ensure some level of accuracy, screening out the really "bad" ratings. Unlimited and anonymous reviews have notoriously poor quality, because a competitor may enter many negative reviews, the seller herself may enter many positive reviews, or someone who has never even used the product or service may enter random reviews. So before anything else, we should first check the mechanism used to enter reviews. Who can enter reviews? How strongly are customers encouraged, or even rewarded, to review? Do you need to enter a review of reviews before you are allowed to upload your own review? Sometimes a seemingly minor change in formatting leads to significant differences: Is it a binary thumbs-up or thumbs-down, followed by a tally of up vs. down votes? What is the dynamic range of the numerical scale? It has been observed that the scale of 1–10 often returns 7 as the average, with a bimodal distribution around it. A scale of 1–3 gives a very different psychological hint to the reviewers compared to a scale of 1–5, or a scale of 1–10 compared to -5 to 5.

Second, the review population size needs to be large enough. But *how* large is large enough? And can we run the raw ratings through some signal processing to get the most useful aggregation?

These are tough questions with no good answers yet, and are not even well-formulated problem statements. The first question depends on the nature of the product being reviewed. Movies (e.g., on IMDb) are very subjective, whereas electronics (e.g., on Amazon) are much less so, with hotels (e.g., on tripadvisor) and restaurants (e.g., on opentable) somewhere in between. It also depends on the quality of the review, although reputation of the reviewer is a difficult metric to quantify in its own right.

The second question depends on the metric of "usefulness." Each user may have a different metric, and the seller of the product or the provider of the service may use yet another one. This lack of clarity in what should be optimized is the crux of the ill-definedness of the problem at hand.

With these challenges, it may feel like opinion aggregation is unlikely to work well. But there have been notable exceptions recorded. A famous example is Galton's 1906 observation on a farm in Plymouth, UK, where 787 people in a festival there participated in a game of guessing the weight of an ox, each writing down a number *independently* of others. There was also no common bias; everyone could take a good look at the ox. While the estimates by each individual were all over the places, the average was 1197 pounds. It turned out the ox weighed 1198 pounds. Just a simple averaging worked remarkably well. For the task of guessing the weight of an ox, 787 was more than enough to get the right answer (within a margin of error of 0.1%).

But in many other contexts, the story is not quite as simple as Galton's experiment. There were several key factors here that made simple averaging work so well.

- The task was relatively easy; in particular, there was an objective answer with a clear numerical meaning.
- The estimates were both unbiased and independent of each other.
- There were enough people participating.

More generally, three factors are important in aggregating individual opinions.

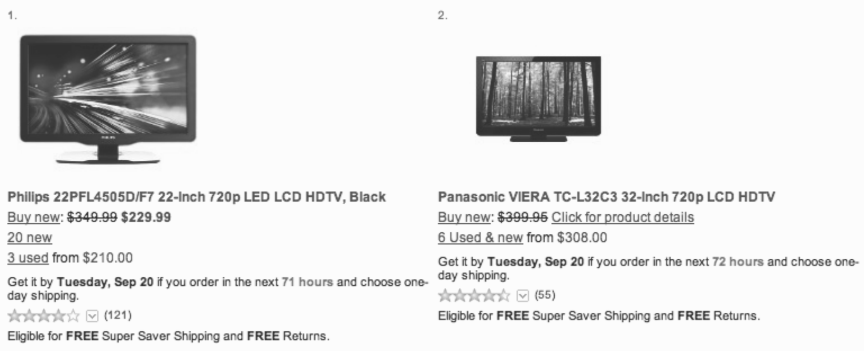
- *Definition of the task*: Guessing a number is easy. Consensus formation in social choice is hard. Reviewing a product on Amazon is somewhere in between. Maybe we can define "subjectivity" by the size of the review population needed to reach a certain "stabilization number."
- *Independence of the reviews*: As we will see, the **wisdom of crowds**, if there is one to a degree we can identify and quantify, stems not from having many smart individuals in the crowd, but from the independence of each individual's view from the rest. Are Amazon reviews independent of each other? Kind of. Even though you can see the existing reviews before entering your own, usually your rating will not be significantly affected by the existing ratings. Sometimes, reviews are indeed entered as a reaction to recent reviews posted on the website, either to counter-argue or to reinforce points made there. This influence from the sequential nature of review systems will be studied in Chapter 7.
- *Review population*: For a given task and the degree of independence, there is correspondingly a minimum number of reviews, a threshold, needed to give a target confidence of trustworthiness to the average. If these ratings pass through some signal-processing filters first, then this threshold may be lowered.

What kind of signal processing do we need? For text reviews, there need to be tools from natural language processing to detect inconsistencies or extreme emotions in a review and to discount it. We in academia face this problem in each decision on a peer-reviewed paper, a funding proposal, a tenure-track position interview, and a tenure or promotion case.

For rating numbers, some kind of weighting is needed, and we will discuss a particularly well-studied one soon. In Chapter 6, we will also discuss voting methods, including majority rule, pairwise comparison, and positional counting. These voting systems require each voter to provide a complete ranking, and sometimes on a numerical rating scale. Therefore, we will have more information, perhaps too much information, as compared with our current problem in this chapter.

5.1.1 Challenges of rating aggregation

Back to rating aggregation. Here are several examples illustrating three of the key challenges in deciding when to trust ratings on Amazon.



1.

Philips 22PFL4505D/F7 22-Inch 720p LED LCD HDTV, Black

Buy new: ~~$349.99~~ **$229.99**

20 new

3 used from $210.00

Get it by **Tuesday, Sep 20** if you order in the next 71 hours and choose one-day shipping.

★★★★☆ (121)

Eligible for **FREE** Super Saver Shipping and **FREE** Returns.

2.

Panasonic VIERA TC-L32C3 32-Inch 720p LCD HDTV

Buy new: ~~$399.95~~ Click for product details

6 Used & new from $308.00

Get it by **Tuesday, Sep 20** if you order in the next 72 hours and choose one-day shipping.

★★★★☆ (55)

Eligible for **FREE** Super Saver Shipping and **FREE** Returns.

Figure 5.1 The tradeoff between review population and average rating score. Should a product with fewer reviews (55) but a higher average rating (4.5 stars) be ranked higher than a competing product with more reviews (121) but a lower average rating (4 stars)?

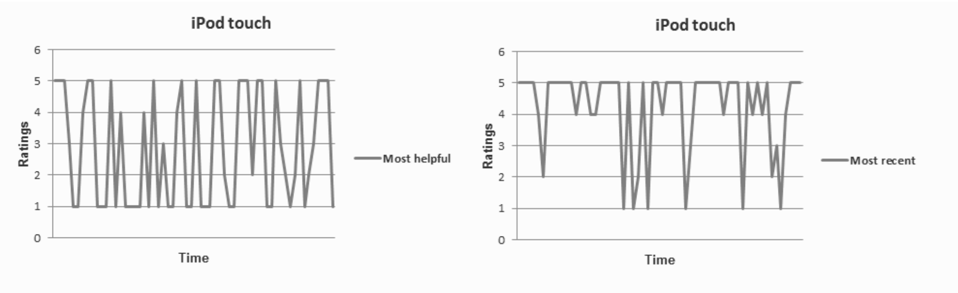


Figure 5.2 How to view the aggregated ratings: should it be based on helpful ratings or on the latest trend? The same set of iPod touch ratings on Amazon is used to extract two different subsets of ratings, and their values are quite different.

Example 1. Many online rating systems use a naive averaging method for their product ratings. Given that different products have different numbers of reviews, it is hard to determine which product has a better quality. For example, as in Figure 5.1, one day in 2011 on Amazon, Philips 22PFL4504D HDTV has 121 ratings with a mean of 4, while Panasonic VIERA TC-L32C3 HDTV has 55 ratings with a mean of 4.5. So the customer is faced with a tradeoff between choosing a product with a lower average rating and a larger number of reviews versus one with a higher average rating and a smaller number of reviews.

Example 2. Consider two speaker systems for home theater on Amazon. However, RCA RT151 and Pyle Home PCB3BK have comparable mean scores around 4. On the one hand, 51.9% of users gave RCA RT151 a rating of 5 stars while 7.69% gave 1 star. On the other hand, 54.2% of users gave 5 stars to Pyle Home PCB3BK while 8.4% gave 1 star. So Pyle Home PCB3BK has not only a higher percentage of people giving it 5 stars, but also a higher percentage of people giving it 1 star. There is a larger *variation* in the ratings of Pyle Home PCB3BK. Does that make the average rating more trustworthy or less?

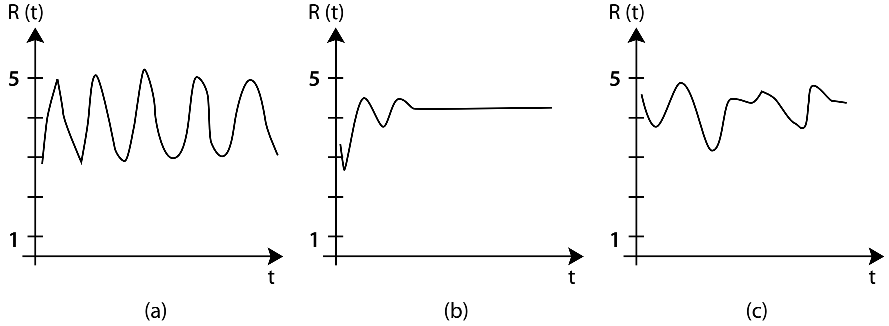


Figure 5.3 Three time series with the same long-term average rating but very different stabilization behaviors. Suppose the time-axis scale is on the order of weeks. Then curve (a) shows continued cyclic fluctuations of ratings over time; curve (b) shows a clear convergence; and curve (c) shows signs of convergence, but it is far from clear that the ratings have converged.

Example 3. In Figure 5.2, we compare the ratings of the top 60 “most helpful” reviews of iPod3 Touch (32 GB) on Amazon with those from the 60 most recent ratings. The mean of the most recent ratings is 1.5 times greater than the mean of the most helpful reviews. Does this reflect a “real” change or just normal fluctuations? What should the timescale of averaging be?

At the heart of these problems is the challenge of turning *vectors* into *scalars*, which we will meet again in Chapter 6. This can be a “lossy compression” with very different results depending on how we run the process of scalarization.

5.1.2 Beyond basic aggregation of ratings

We may run a time-series analysis to understand the dynamics of rating. In Figure 5.3, the three curves of ratings entered over a period of time give the same average, but “clearly” some of them have not converged to a stable average rating. What kind of *moving-window size* should we use to account for cumulative averaging and variance over time?

We may consider detecting anomalous ratings and throwing out the highly suspicious ones. If we detect a trend change, that may indicate a change of ownership or generational upgrade. And if such detection is accurate enough, we can significantly discount the outdated ratings. For ratings on the scale of $1-5$, the coarse granularity makes this detection more difficult.

We may consider zooming into particular areas of this vector of ratings, e.g., the very satisfied customers and the very dissatisfied ones, although it is often the case that those who care enough to enter ratings are either extremely satisfied or reasonably dissatisfied. There might be a bimodal distribution in the underlying customer satisfaction for certain products, but for many products there is often another bimodal distribution on the biased sampling since only those who cared enough to write reviews.

Across all these questions, we can use the cross-validation approach from Chapter 4 to train and test the solution approach. Or, if we can stand back one year and predict the general shape of ratings that have unfolded since then; that would be a strong indicator of the utility of our signal-processing method.

But these questions do not have well-studied answers yet, so we will now focus instead on some simpler questions as proxies to our real questions: Why does simple averaging sometimes work, and what should we do when it does not?

5.2 A Long Answer

5.2.1 Averaging a crowd

We start from a significantly simplified problem. Take the Galton example, and say the number that a crowd of N people wants to guess is x, and each person i in the crowd makes a guess y_i:

$$y_i(x) = x + \epsilon_i(x),$$

i.e., the true value plus some error ϵ_i. The error depends on x but not on other users j; it is independent of other people's errors. This error can be positive or negative, but we assume that it averages across different x to be 0; it has no bias. In reality, errors are often neither independent nor unbiased. Sequential estimates based on publicly announced estimates made by others may further exacerbate the dependence and bias. We will see examples of such information cascades in Chapter 7.

We measure error by the metric of mean squared error (MSE), just like what we did for Netflix recommendation in Chapter 4. We want to compare the following two quantities:

- the average of individual guesses' errors, and
- the error of the averaged guess.

The average of errors and the error of the average are not the same, and we will see how much they differ. Since x is a number that can take on different values with different probabilities, we should talk about the *expected MSE*, where the expectation $\mathbf{E}_x$ is the averaging procedure over the probability distribution of x.

The average of (expected, mean-squared) errors, denoted by AE, by definition, is

$$E_{AE} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{E}_x\left[\epsilon_i^2(x)\right]. \tag{5.1}$$

On the other hand, the (expected, mean-squared) error of the average, denoted by EA, is

$$E_{EA} = \mathbf{E}_x\left[\left(\frac{1}{N}\sum_{i=1}^{N}\epsilon_i(x)\right)^2\right] = \frac{1}{N^2}\mathbf{E}_x\left[\left(\sum_{i=1}^{N}\epsilon_i(x)\right)^2\right], \tag{5.2}$$

6 Why does Wikipedia even work?

Now we move from recommendation to influence in social networks. We start with consensus formation from conflicting opinions in this chapter before moving on to a collection of influence models in the next two.

But first, let us compare the four different "consensus" models we covered in Chapters 3 – 6, as visualized in Figure 6.1.

- Google's PageRank turns a graph of webpage connections into a single rank-ordered list according to their importance scores.
- Netflix's recommendation turns a user–movie rating matrix into many ranked order lists, one list per user, based on the predicted movie ratings for each user.
- Amazon's rating aggregation turns a vector of rating scores into a single scalar for each product.
- Voting systems turn a set of rank-ordered lists into a single rank-ordered list, as we will see in this chapter.

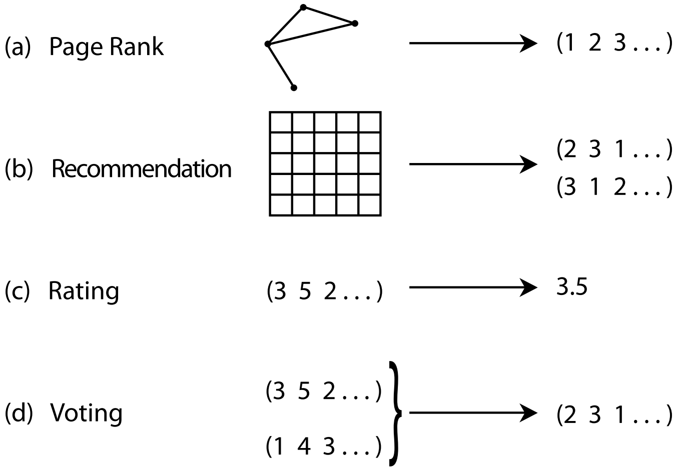


Figure 6.1 Comparison of four types of "consensus-formation" mechanisms with their inputs and outputs. The first three mechanisms have been covered in the last three chapters, and the last one will be part of this chapter.

6.1 A Short Answer

Crowdsourcing knowledge representation with unpaid and possibly anonymous contributors is very tricky. It faces many challenges for this idea to “work.” For example, how do we create incentives for people to keep contributing; and how do we handle disagreements among the contributors?

Launched in 2001, Wikipedia represented a convergence of three forces that had been gathering momentum: (1) wikis for online collaboration among people, (2) the free- and open-software movement, and (3) the appearance of online encyclopedias. Within a decade, Wikipedia has generated 4 million articles in the USA and 27 million articles worldwide. It has become one of the most popular sources of information online. For certain fields, like medicine, the quality of Wikipedia articles is consistently high; and for many fields, if you google a term, a Wikipedia entry will likely come up in the top few search results. It is quite amazing that Wikipedia actually “worked” as well as it did. As we have seen in Chapter 1 and will see again in Chapter 11, when people interact with each other, there is often the risk of the “tragedy of the commons”. How does Wikipedia turn that into effective collaboration? This is the driving question for this chapter.

Of course, there are also limitations to Wikipedia in its capacity as an encyclopedia.

- *Misinformation*: Sometimes information on Wikipedia is plainly wrong, especially in articles with a small audience. But Wikipedia provides an effective self-correcting mechanism: it is open to edits from anyone.
- *Mistakes*: There are also honest mistakes, but, again, anyone can edit an article, and the edit will stay there as long as no other contributor can present a stronger case otherwise.
- *Missing information*: No encyclopedia can be truly *complete* to everyone’s liking, not even the largest encyclopedia in history.

Due to these limitations, there have been some high-profile cases of abuse in Wikipedia. Still, Wikipedia stands as a major success of online collaboration.

There had been other efforts aimed at creating free, open, online encyclopedias before, and the unique success of Wikipedia is often attributed to a “good-faith collaboration” environment within the Wikipedia contributor community. If we count the number of pairwise links in a fully connected graph with n nodes, we have on the order of n^2 such links. But if we examine the number of opinion configurations, we have 2^n possibilities even if each person has just two choices. This n^2 vs. 2^n tension exemplifies the positive and the negative sides of the network effect. Converging on one of these 2^n configurations is difficult, and Wikipedia mostly follows the principle of “rough consensus.” In this chapter, we will study the process of reaching a rough consensus from voting theory (even though it does not explicitly involve voting through rank-ordered lists) and from bargaining theory.

Wikipedia is free, open, dynamic, interactive, and extensively linked. There are natural pros and cons associated with such a model of an encyclopedia that complements other forms of encyclopedia. Let us consider three distinct features of Wikipedia.

- It is free. How can people be motivated to contribute? Incentives do not have to be financial; the ability to influence others is a reward in its own right to most people. This requires the Wikipedia audience to be very large.
- Anyone can write or add to an article, including non-experts, anonymous writers, and people with conflicts of interest. The key is *check and balance.* Precisely because anyone can contribute, Wikipedia has a large body of writers who check others' writing frequently through a mechanism for debates and updates. Sometimes, however, a contributor or an IP address may be blocked if it is detected as a regular source of deliberate misinformation.
- Any subject may be contributed, including controversial ones. Sometimes, however, certain articles can be "protected" from too frequent edits to give time for the community of contributors to debate. How does Wikipedia avoid unbalanced treatment or trivial subjects? It turns out that there are Policies and Guidelines, and there are mechanisms for conflict resolution by editors.

The first and second features above provide Wikipedia with a strong, positive network effect: a larger audience leads to more contributors, which in turn leads to more audience, provided that the quality of contributions is kept high.

This brings us to the third feature above. How does Wikipedia enforce quality and resolve conflicting contributions? (Before addressing this question, we must bear in mind the obvious fact that Wikipedia is not a sovereign state with the power of a government. So issues such as voting, decision-making, and free speech do not have the same context.)

To start with, there are three core Policies on Wikipedia to help ensure reliability and neutrality of the articles as much as possible.

- *Verifiability (V)*: each key point and all data in an article must be externally verifiable, with a link to the primary source for verification by readers.
- *No Original Research (NOR)*: this is to prevent people from using Wikipedia as a publication venue of their new results.
- *Neutral Point of View (NPOV)*: the basic rule is that a reader must not be able to tell the bias of the author in reading through a Wikipedia article. It is particularly important for controversial topics, but also the most difficult to use exactly in those cases, e.g., contentious political, social, and religious topics. Unlike the above two policies, it is harder to enforce this one since "neutrality" is subjective.

Wikipedia has also installed several mechanisms for debates and updates. One is the use of the history page and the talk page, which are available for public view through tags on top of each article's page. All previous versions and all the changes made are recorded.

Furthermore, there is a reputation system for contributors, similar to the reviewer rating system on Amazon. Each article can be rated on a 1–6 scale. For those who do not reveal their names, it is a reputation system of the IP addresses of the devices from which contributions are sent. In addition, links across article pages are analyzed in a manner similar to Google's PageRank.

But perhaps the ultimate mechanism still boils down to people negotiating. Depending on the stage of the article, expert and non-expert contributors may join the discussion. There is a hierarchy of Wikipedia communities, and debates among contributors who cannot come to an agreement will be put forward to a group of the most experienced editors. This committee of editors acts like a jury, listening to the various sides of the debate, and then tries to decide by "rough consensus."

How do we model the process of reaching a "rough consensus" through "good faith collaboration?" Not easy. We will see in this chapter two underlying theories: voting theory and bargaining theory. But much translation is needed to connect either of these theories to the actual practice of Wikipedia.

- In a voting model, each contributor has some partially ordered list of preferences, and a *threshold* on how far away from her own preferences the group decision can be before she vetoes the group decision and thereby preventing the consensus from being reached. (Sometimes a decision is actually carried out by explicit votes in the arbitration committee. And the committee members are also elected through a voting system.) Dealing with partial ordering, quantifying the distance between two ordered lists, and modeling each editor's veto-decision threshold are still under-explored in the study of group-interaction dynamics in Wikipedia.
- In a bargaining model, the contributors need to reach a compromise, otherwise there would be no agreement. Each contributor's utility function, and the default position in the case of no agreement, need to reflect the goodwill typically observed in Wikipedia collaboration.

In contrast to coordination through well-defined pricing feedback signals, which we will see in several later chapters, coordination though bargaining or voting is much harder to model. In the next section, we will present the basics of the rich theory of voting and social choice. Then, in the Advanced Material, we will briefly discuss the mathematical language for bargaining and cooperative games.

6.2 A Long Answer

Wikipedia consensus formation illustrates important issues in reaching consensus among a group of individuals that is binding for everyone. This is different from presenting the rank-ordered list for each person to evaluate individually

7 How do I viralize a YouTube video and tip a Groupon deal?

A quick recap of where we have been so far in the space of online services and web 2.0. In Chapter 3, we discussed the recommendation of webpages with an objective metric computed by Google from the graph of hyperlinked webpages. In Chapter 4, we discussed the recommendation of movies with subjective opinions estimated by Netflix from movie–user bipartite graphs.

Then we investigated the wisdom of crowds. In Chapter 5, we discussed *aggregation* of opinion in (more or less) independent ratings on Amazon. In Chapter 6, we discussed *resolution* of opinion conflicts in Wikipedia.

In this chapter, we will talk about *dependence* of opinions, taking a macroscopic, topology-agnostic approach, and focusing on the viral effect in YouTube and tipping in Groupon. Then in the next chapter, we will talk about the effect of network topology on the dependence of opinion.

As will be further illustrated in this and the next chapters, network effects can be positive or negative. They can also be studied as *externalities* (e.g., coupling in the objective function or the constraint functions, where each user's utility or constraint depends on other users' actions), or as *information dependence* (e.g., information cascades or product diffusion as we will see in this chapter).

7.1 A Short Answer

7.1.1 Viralization

YouTube is a "viral" phenomenon itself. In the space of user-generated video content, it has become the dominant market leader, exhibiting the "winner takes all" phenomenon. More recently it has also featured movies for purchase or rental, and commissioned professional content, to compete against Apple's iTunes and the studios.

YouTube started in February 2005 and was acquired by Google in 2006. Within several years people watched videos on YouTube so much that it became the second largest search engine with 2.6 billion searches in August 2008, even though we normally would not think of YouTube as a search engine. Its short video-clip format, coupled with its recommendation page, is particularly addictive. In summer 2011, more than 40% of Internet videos were watched on YouTube, with over 100 million unique viewers each month just in the USA. Each day over a billion video plays were played, and each minute more than 24 hours of new video clips were uploaded.

There are interesting analytic engines like "YouTube Insight" that highlight the aggregate behavior of YouTube watching. Some videos have gone viral, the most extreme example being "Charlie bit my finger–again," a less-than-one-minute clip that had generated over 465 million views as of July 2012. If you just look at the viewer percentage across all the videos on YouTube, it exhibits the long-tail distribution that we will discuss in Chapter 10.

There has been a lot of social media and web 2.0 marketing research on how to make your YouTube video go viral, including practical advice on the four main paths that lead a viewer to a YouTube clip:

- web search,
- referral through email or twitter,
- subscription to a YouTube channel, and
- browsing through the YouTube recommendation page.

We have seen how tags, subscription, and recommendation play a bigger role than the counts of likes and dislikes in the rise of popularity of a video. It is also interesting to see that YouTube does not use a PageRank-style algorithm for ranking the video clips, since linking the videos by tags is too noisy. Nor does it use the sophisticated recommendation engine such as Netflix Prize solutions, since viewing data for short clips is too noisy and YouTube videos often have short lifecycles. Instead, YouTube recommendation simply leverages video association through **co-visitation count**: how often each pair of videos is watched together by a viewer over, say, 24 hours. This gives rise to a set of related videos for each video, a link relationship among the videos, and thus a graph with the videos as nodes. From the set of videos in k hops from a given video, together with matching of the keywords in the video title, tags, and summary, YouTube then generates a top-n recommendation page. It has also been observed that often only those videos with a watch-count number similar to, or slightly higher than, that of the current video are shown in the recommendation page. This is a version of "preferential attachment" that we will discuss in Chapter 10. It makes it easier for widely watched videos to become even more widely watched, possibly becoming viral.

Now, how do you even define "viral"? There is no commonly accepted definition, but probably the notion of "viral" means that the rate-of-adoption curve should exhibit three features, like curve (c) shown in Figure 7.1:

- high peak,
- large volume, i.e., the adoption lasts long enough in addition to having a high peak, and
- a short time to rise to the peak.

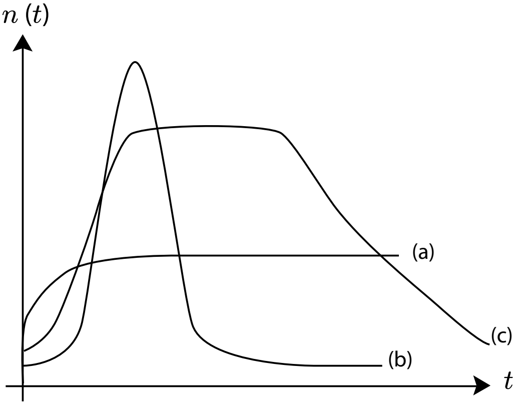


Figure 7.1 A few typical shapes of adoption trajectory $n(t)$ over time t: curve (a) stays at a low level; curve (b) rises very quickly but then dies out rapidly too; and curve (c) has a reasonably sharp rise to a high level and a large area under the curve. Of course, we can also have combinations of these curves, e.g., a curve that has a sharp rise to a high level and stays there.

7.1.2 Tipping

Another web 2.0 sensation that has gone viral is the daily deal service, such as Groupon and LivingSocial. Groupon was formed in 2008, and after two years was generating over \$300 million annual revenue from more than 500 cities. It went pubic in November 2011.

In a daily deal, a supplier of some goods or services announces a special discount, which must have a large enough number of users signed up within a 24-hour period. If the number of users exceeds the target threshold, the deal is tipped, and each user has, say, 3 months to redeem the coupon. The supplier's hope is that the discount is in part compensated for by the high volume, and in part the possibility of repeat customers who will return in the future and pay the full price. This is the **power of crowds** in action. More specifically, a daily-deal tipping needs a sufficiently large number of people to make the same decision within a sufficiently small window of time.

The effectiveness of Groupon (and the like) for the suppliers and the consumers is still somewhat under-studied. In a detailed survey by Dhulakia in 2011 with 324 businesses in 23 US market, some interesting results emerged: close to 80% of deal users were new customers, but only 36% of them spent beyond the deal's face value, and only 20% returned to buy at full price later. On the supplier side, 55% made money from the deals, but 27% lost money, and less than half of them expressed interest in participating in the future, restaurants and salons being particularly negative. Across hundreds of daily-deals websites, there are few differentiation factors at this point. The cut into the deals' revenues by these websites will have to be lowered in the future as the industry consolidates.

This chapter presents models that can be used to characterize and understand phenomena such as viralization, tipping, and synchronization observed for YouTube videos and Groupon deals.

7.2 A Long Answer

There are two distinct reasons for popularity of a product.

1. *Intrinsic value*: You may enjoy certain music or buy a particular product just because you like it, whether the rest of the world agrees or not.
2. *Network effect*: Your decision depends on what others do, either because (a) the fact that others like a product gives you information, possibly leading to what we call information cascades, an example of the fallacy of crowds, or (b) because the value of the service or product actually depends on the number of people who use it, like the fax machine or Wikipedia, an example of positive externality.

We will first discuss models of 2(a), such as **information cascade** studied in the political-economy literature and **tipping** from an unstable equilibrium towards a stable equilibrium. Then, in the Advanced Material, we will discuss the combination of 1 and 2(b), called the intrinsic factor and the imitation factor in **diffusion** models studied in the marketing literature, as well as a **synchronization** model.

All the models in this chapter are population-based and agnostic of the actual topologies (although our example of information cascade implicitly assumes a linear topology). In the next chapter, we will focus on topology-based models in the study of influence.

The models in both chapters are summarized in Table 7.2. Except for the synchronization and random-walk models, all assume the nodes (the people) have discrete, often binary, "states of mind." We will also see that some of these models become similar when generalized a little.

Which model to use really depends on what we are trying to model. Many people acting at the same time? Or a few early adopters changing others' minds? Or one more person carrying the system over the threshold and triggering a change in others' behavior? Each of these models is motivated by a different type of influence, and has its use and limitation.

Viralizing a YouTube video, tipping a Groupon deal, and influencing via Facebook and Twitter posts are three particular examples in these two chapters. But, for these emerging phenomena involving human-psychology factors, we still do not know much about which of these models and their extensions, if any, fit reality sufficiently well to render predictive power.

Across these models, the following issues are raised and some of them have been addressed:

8 How do I influence people on Facebook and Twitter?

To study a network, we have to study both its *topology* (the graph) and its *functionalities* (tasks carried out on top of the graph). This chapter on topology-dependent influence models does indeed pursue both, as do the next two chapters.

8.1 A Short Answer

Started in October 2003 and formally founded in February 2004, Facebook has become the largest social network website, with 900 million users worldwide as of spring 2012 at the time of its IPO. Many links have been formed among these nodes, although it is not straightforward to define how many mutual activities on each other's wall constitute a "link."

Founded in July 2006, Twitter attracted more than 500 million users in six years. At the end of 2011, over 250 million tweets were handled by Twitter each day. Twitter combines several functionalities into one platform: microblogging (with no more than 140 characters), group texting, and social networking (with one-way following relationships, i.e., directional links).

Facebook and Twitter are two of the most influential communication modes, especially among young people. For example, in summer 2011's east-coast earthquake in the USA, tweets traveled faster than the earthquake itself from Virginia to New York. They have also become a major mechanism in social organization. In summer 2009, Twitter was a significant force in how the Iranians organized themselves against the totalitarian regime.

There have been all kinds of attempts at figuring out

- (1) how to quantify the statistical properties of opinions on Facebook or Twitter;
- (2) how to measure the influential power of individuals on Facebook or Twitter; and
- (3) how to leverage the knowledge of influential power's distribution to actually influence people online.

For example, on question (1), our recent study of all the tweets about Oscar-nominated movies during the month of February 2012 shows a substantial skew towards positive opinion relative to other online forums, and the need to couple tweet analysis with data from other venues like the IMDb or Rotten Tomato to get a more accurate reading of viewer reception and prediction of box office success.

Question (2) is an analysis problem and question (3) a synthesis problem. Neither is easy to answer; and there is a significant gap between theory and practice, perhaps the biggest such gap you can find in this book. Later in this chapter, we will visit some of the fundamental models that have yet to make a significant impact on characterizing and optimizing influence over these networks.

But the difficulty did not prevent people from trying out heuristics. Regarding (2), for example, there are many companies charting the influential power of individuals on Twitter, and there are several ways to approximate that influential power: by the number of followers, by the number of retweets (with "RT" or "via" in the tweet), or by the number of repostings of URLs. There are also many companies data-mining the friendship network topology of Facebook.

As to (3), Facebook uses simple methods to recommend friends, which are often based on email contact lists or common backgrounds. Marketing firms also use Facebook and Twitter to stage marketing campaigns. Some "buy off" a few influential individuals on these networks, while others buy off a large number of randomly chosen, reasonably influential individuals.

It is important to figure out who the influential people are. An often-quoted historical anecdote concerns the night rides by Paul Revere and by William Dawes on 18-19 April in 1775. Dawes left Boston earlier in the evening than did Revere. They took different paths towards Lexington, before riding together from Lexington to Concord. Revere alerted influential militia leaders along his route to Lexington, and was therefore much more effective in spreading the word of the imminent British military action. This in turn lead to the American forces winning on the next day the first battle that started the American Revolutionary War.

How do we quantify which nodes are more important? The question dates back thousands of years, and one particularly interesting example occurred during the Renaissance in Italy. The Medici family was often viewed as the most influential among the fifteen prominent families in Florence during the fifteenth and sixteenth centuries. As shown in Figure 8.1, it sat in the "center" of the family social network through strategic marriages. We will see several ideas quantifying the notion of centrality.

How do we quantify which links (and paths) are more important? We will later define strong vs. weak ties. Their effects can be somewhat unexpected. For example, Granovetter's 1973 survey in Amherst, Massachusetts showed the strength of weak ties in spreading information. We will see another surprise of weak ties' roles in social networks, on six-degree separation in Chapter 9.

Furthermore, how do we quantify which subset of nodes (and the associated links) are connected enough among themselves, and yet disconnected enough from the rest of the network, that we can call them a "group"? We save this question for the Advanced Material.

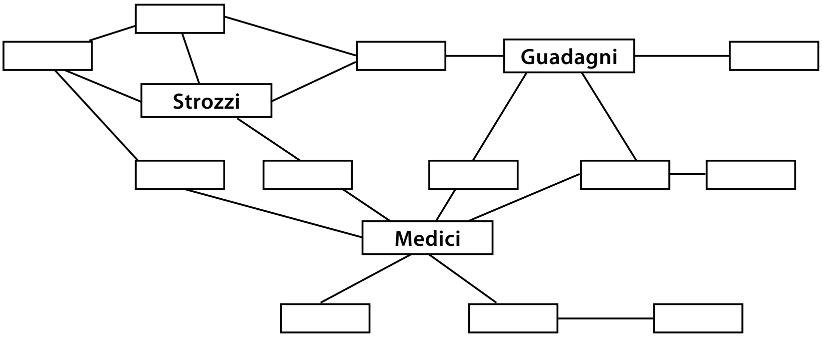


Figure 8.1 Padgett's Florentine-family graph shows the central position of the Medici family in Renaissance Florence. Each node is a family, three of them shown with their names. Each link is a marriage or kinship relationship. The Medici family clearly had the largest degree, but its influential power relative to the other families was much more than the degree distribution would indicate. Other measures of centrality, especially betweenness centrality, reveal just how influential the Medici family was.

8.1.1 Graphs and matrices

Before proceeding further, we first formally introduce two commonly used matrices to describe graphs. We have seen a few different types of graphs in the previous chapters. In general, a graph G is a collection of two sets: V is the set of vertices (nodes) and E is the set of edges (links). Each link is in turn a directed two-tuple: the starting and the ending nodes of that link.

We will construct a few other matrices later as concise and useful representations of graphs. We will see that properties of a graph can often be summarized by linear-algebraic quantities about the corresponding matrices.

The first is the **adjacency matrix A**, of dimension $N \times N$, of a given graph $G = (V, E)$ with N nodes connected through links. For the graphs we deal with in this book, A_{ij} is 1 if there is a link from node i to j, and 0 otherwise. We mostly focus on **undirected graphs** in this chapter, where each link is **bidirectional**. Given an undirected graph, **A** is symmetric: $A_{ij} = A_{ji}, \forall i, j$. If a link can be **unidirectional**, we have a **directed graph**, like the Twitter following relationship graph.

The second, which is less used in this book, is the **incidence matrix $\hat{\mathbf{A}}$**, of dimension $N \times L$, where N is again the number of nodes and L the number of links. For an undirected graph, $\hat{A}_{ij} = 1$ if node i is on link j, and 0 otherwise. For a directed graph, $\hat{A}_{ij} = 1$ if node i starts link j, $\hat{A}_{ij} = -1$ if node i ends link j, and $\hat{A}_{ij} = 0$ otherwise.

Straightforward as the above definitions may be, it is often tricky to define what exactly constitutes a link between two persons: being known to each other by first-name as in Milgram's small-world experiment? Or "friends" on Facebook who have never met or communicated directly? Or only those to whom you text at least one message a day? Some links are also directional: I may have commented on your wall postings on Facebook but you never bothered reading my wall at all. Or I may be following your tweets, but you do not follow mine.

Even more tricky is to go beyond the simple static graph metrics and into the functionalities and dynamics on a graph. That is a much tougher subject. So we start with some simple static graph metrics first.

8.2 A Long Answer

8.2.1 Measuring node importance

You may be in many social networks, online as well as offline. How important are you in each of those networks? Well, that depends on how you define the “importance” of a node. It depends on the specific functionalities we are looking at, and it evolves over time. But we shall restrict ourselves to just static graph metrics for now. Neither is it easy to discover the actual topology of the network. But let us say for now that we are given a network of nodes and links.

There are at least four different approaches to measuring the importance, or **centrality**, of a node, say node 1.

The first obvious choice is **degree**: the number of nodes connected to node 1. If it is a directed graph, we can count two degrees: the in-degree: the number of nodes pointing towards node 1, and the out-degree: the number of nodes that node 1 points to. **Dunbar’s number**, usually around 150, is often viewed as the number of friends a typical person may have, but the exact number of course depends on the definition of “friends.” The communication modes of texting, tweeting, and blogging may have created new shades of definition of “friends.” In Google+, you can also create your own customized notions of friends by creating new circles.

We will see there are many issues with using the degree of a node as its centrality measure. One issue is that if you are connected to more-important nodes, you will be more important than you would be if you were connected to less-important nodes. This may remind you of PageRank in Chapter 3. Indeed, we can take PageRank’s importance scores as a centrality measure.

A slightly simpler but still useful view of centrality is to just look at the successive multiplication of the centrality vector $\mathbf{x}$ by the adjacency matrix $\mathbf{A}$ that describes the network topology, starting with an initialization vector $\mathbf{x}[0]$:

$$\mathbf{x}[t] = \mathbf{A}^t\mathbf{x}[0].$$

In a homework problem, you will discover a motivation for this successive multiplication.

We can always write a vector as a linear combination of the eigenvectors $\{\mathbf{v}_i\}$ of $\mathbf{A}$, arranged in descending order of the corresponding eigenvalues and indexed

9 Can I really reach anyone in six steps?

In the last chapter, we saw the importance of topology to functionality. In this and the next chapters, we will focus on **generative models** of network topology and reverse-engineering of network functionality. These are mathematical constructions that try to *explain* widespread empirical observations about social and technological networks: the "small world" property and the "scale free" property. We will also highlight common misunderstandings and misuse of generative models.

9.1 A Short Answer

Since Milgram's 1967 experiment, the **small world** phenomenon, or the **six degrees of separation**, has become one of the most widely told stories in popular science books. Milgram asked 296 people living in Omaha, Nebraska to participate in the experiment. He gave each of them a passport-looking letter, and the destination was in a suburb of Boston, Massachusetts, with the recipient's name, address, and occupation (stockbroker) shown. Name and address sound obvious, and it turned out that it was very helpful to know the occupation. The goal was to send this letter to one of your friends, defined as someone you knew by first name. If you did not know the recipient by first name, you had to send the letter via others, starting with sending it to a friend (one hop), who then sent it to one of her friends (another hop), until the letter finally arrived at someone who knew the recipient by first name and sent it to the recipient. This is illustrated in Figure 9.1.

Of these letters, 217 were actually sent out and 64 arrived at the destination, a seemingly small arrival rate of 29.5% but actually quite impressive, considering that a later replica of the experiment via email had only a 1.5% arrival rate. The other letters might have been lost along the way, and needed to be treated carefully in the statistical analysis of this experiment's data. But, out of those 64 that arrived, the average number of hops was 5.2 and the median 6, as shown in Figure 9.2.

Researchers have long suspected that the **social distance**, the average number of hops of social relationships it takes (via a short path) to reach anyone in a population, grows very slowly as the population size grows, often logarithmically.

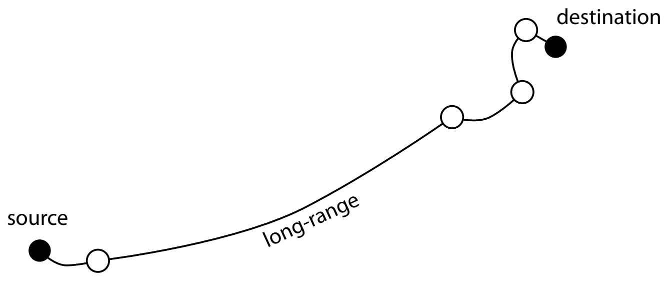


Figure 9.1 A picture illustrating the Milgram experiment in 1967. A key phenomenon is that there is often one or two long-range links in these short paths between Omaha and Boston. It turns out that the long-range links substantially reduced the shortest paths' and the searchable paths' lengths without reducing the clustering coefficient significantly in the social network.

Milgram's celebrated experiment codified the viewpoint. From the 1970s to the online social media era, much empirical evidence suggested the same: from the Erdos number among mathematicians to co-starring relationships in the IMDb.

Should we be surprised by this seemingly universal observation of social networks? There are two issues here, echoing the dichotomy between topology and functionality.

- One is *structural*: There exist short paths in social networks.
- Two is *algorithmic*: With very limited local information a node can navigate through a social network and find a short path to a given destination.

The second kind of small worlds is more surprising than the first and requires more careful modeling of the functionality of **social search**. For example, a report in November 2011 computed the degrees of separation on Facebook to be 4.74. That concerned only with the existence of short paths, not the more relevant and more surprising discoverability of short paths from local information. As we will see, it is also more difficult to create a robust explanation for the observation of an algorithmic small world.

But first, we focus on the existence of short paths. On the surface, it seems fascinating that you can likely reach anyone in six steps or fewer. Then, on second thoughts, you may reason that, if I have twenty friends, and each of them has twenty friends, then in six steps, I can reach 20^6 people. That is already 64 million people. So of course six steps often suffice.

But then, giving it further thought, you realize that social networks are filled with"triangles," or **triad closures**, of social relationships. This is illustrated in Figure 9.3: if Alice and Bob both know Chris, Alice and Bob likely know each other directly too. This is called **transitivity** in a graph (not to be confused with transitivity in voting). In other words, the catch of the 20^6 argument above is that you need your friend's friends to not overlap with your own friends. Otherwise,

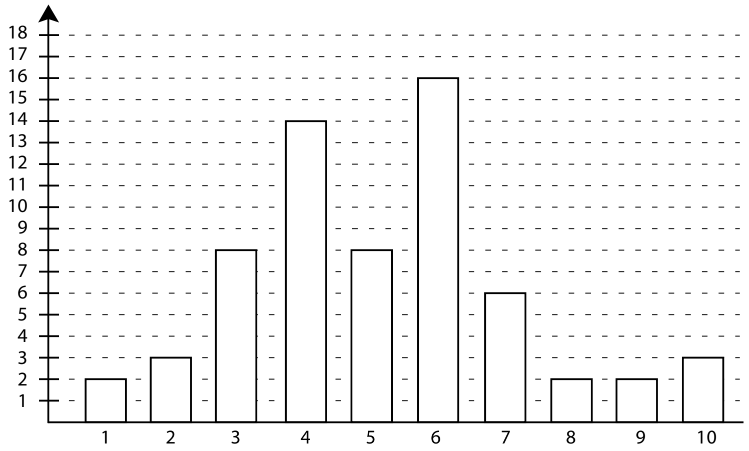


Figure 9.2 The histogram of the length of the search paths from different sources in Omaha to the common destination in Boston in Milgram's experiment. The median value is six, leading to the famous six degrees of separation.

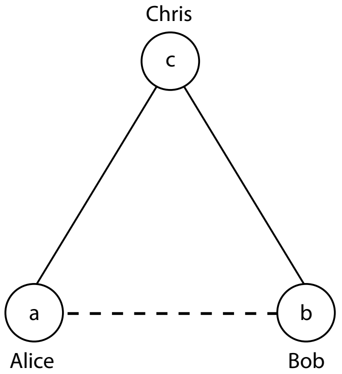


Figure 9.3 An illustration of triad closure in social networks. If Alice knows Chris and Bob knows Chris (the solid lines), it is likely that Alice and Bob also know each other (the dotted line). If so, the connected triple forms a triangle. The clustering coefficient quantifies the ratio between connected triples and triangles in a graph.

the argument fails. But of course, many of your friend's friends are your own friends too. There is a lot of overlap. The phenomenon of people who are alike tending to form social links is called **homophily**, and it can be quantified by the **clustering coefficient** as discussed later. Now, six degrees of separation is truly surprising.

Milgram-type experiments suggest something even stronger: not only are there short paths, but they can be discovered by each individual node using very limited information about the destination and its local view of the network.

Compared with routing packets through the Internet, social search is even harder since nodes do not pass messages around to help each other construct some global view of the network topology. That help is implicitly embedded in the address and occupation of the recipient, and possibly in the name that can reveal something about the destination's sex and ethnicity. Some kind of *distance metric* must have been constructed in each person's mind throughout Milgram's experiment. For example, New York is closer to Boston than Chicago is, on the geographic proximity scale measured in miles. Or, a financial advisor is perhaps closer to a stockbroker than a nurse is, on some occupation proximity scale, which is more vague but nonetheless can be grossly quantified. Suppose each person uses a simple, "greedy" algorithm to forward the letter to her friend who is closest to the destination, where "closeness" is defined by a composite of these kinds of distance metrics. Is it a coincidence that this social search strategy discovers a short path?

We will walk through several models that address the above issues.

9.2 A Long Answer

9.2.1 Structural small worlds: Short paths

There are several ways to measure how "big" a (connected) graph is. One is diameter: it is the length of the longest shortest path between any pair of nodes. "Longest shortest" may sound strange. Here, "shortest" is with respect to all the paths between a given pair of nodes, and "longest" is with respect to all possible node pairs.

When we think of a network as a small world, however, we tend to use the *median* of the shortest paths between all node pairs, and look at the growth of that metric as the number of nodes increases. If it grows at a rate on the order of the log of the number of nodes, we say the network is (structurally) a small world.

Suppose we are given a fixed set of n nodes, and for each pair of nodes decide with probability p that there is a link between them. This is the basic idea of a **Poisson random graph**, or the **Erdos–Renyi model**.

Of course, this process of network formation does not sound like most real networks. It turns out that neither does it provide the same structures as those we encounter in many real networks. For example, while in a random graph the length of the average shortest path is small, it does not have the right clustering coefficient. For a proper explanatory model of small world networks, we need the shortest path's length to be small and the clustering coefficient to be large.

What is the clustering coefficient? Not to be confused with the density of a cluster from Chapter 8, it is a metric to quantify the notion of triad closure. As in Figure 9.3, we define a set of three nodes (a, b, c) in an undirected graph as a **connected triple**, if there is a path connecting them.

10 Does the Internet have an Achilles' heel?

10.1 A Short Answer

It does not.

10.2 A Long Answer

10.2.1 Power-law distribution and scale-free networks

Sure, the Internet has many security loopholes, from cyber-attack vulnerability to privacy-intrusion threats. But it does not have a few highly-connected routers in the center of the Internet that an attacker can destroy to disconnect the Internet, which would have fit the description of an "Achilles' heel". So why would there be rumors that the Internet has an Achilles' heel?

The story started in the late 1990s with an inference result: the Internet topology exhibits a **power-law distribution** of node degrees. Here, the "topology" of the Internet may mean any of the following:

- the graph of webpages connected by hyperlinks (like the one we mentioned in Chapter 3),
- the graph of Autonomous Systems (ASs) connected by the physical and business relationships of peering (we will talk more about that in Chapter 13), and
- the graph of routers connected by physical links (the focus of this chapter).

For the AS graph and the router graph, the actual distribution of the node degrees (think of the histogram of the degrees of all the nodes) is not clear due to measurement noise. For example, the AS graph data behind the power-law distribution had more than 50% of links missing. Internet exchange points further lead to many peering links among ASs. These are "shortcuts" that enable settlement-free exchange of Internet traffic, and cannot be readily measured using standard network-layer measurement probes.

To talk about the Achilles' heel of the Internet, we have to focus on the graph of routers as nodes, with physical links connecting the nodes. No one knows for sure what that graph looks like either, so people use proxies to estimate it through measurements like trace-route. Studies have shown that such estimates

11 Why do AT&T and Verizon Wireless charge me $10 a GB?

11.1 A Short Answer

Almost all of our utility bills are based on the amount we consume: water, electricity, gas, etc. But even though wireless cellular capacity is expensive to provide and difficult to crank up, consumers in some countries like the USA have been enjoying flat-rate buffets for mobile Internet access for many years. Can a restaurant keep offering buffets with the same price if its customers keep doubling their appetites every year? Or will it have to stop at some point?

In April 2010, AT&T announced its usage-based pricing for 3G data users. This was followed in March 2011 by Verizon Wireless for its iPhone and iPad users, and in June 2011 for all of its 3G data users. In July 2011, AT&T started charging fixed broadband users on U-Verse services on the basis of usage too. In March 2012, AT&T announced that those existing customers on unlimited cellular data plans will see their connection speeds throttled significantly once the usage exceeds 5 GB, effectively ending the unlimited data plan. The LTE data plans from both AT&T and Verizon Wireless for the “new iPad” launched soon after no longer offered any type of unlimited data options. In June 2012, Verizon Wireless updated their cellular pricing plans. A customer could have unlimited voice and text in exchange for turning an unlimited data plan to usage-based. AT&T followed with a similar move one month later. What a reversal going from limited voice and unlimited data to unlimited voice and limited data. Similar measures have been pursued, or are being considered, in many other countries around the world for 3G, 4G, and even wired broadband networks.

How much is 1 GB of content? If you watch 15 minutes of medium-resolution YouTube videos a day, and do nothing else with your Internet access, that is about 1 GB a month. If you stream one standard-definition movie, it is about 2 GB. With the proliferation of capacity-hungry apps, high-resolution video content, and cloud services (we will discuss cloud and video networking in Chapters 16 and 17, respectively), more users will consume more GBs as months go by. This year’s heavy users will become a “normal” user in just a couple of years’ time. With the 4G LTE speed much higher than that of 3G (we will look into the details of speed calculation in Chapter 19), many of these GBs will be consumed on mobile devices and fall into the $10/GB bracket. Those who are used to flat-rate, buffet-style pricing will naturally find this quite annoying. And if

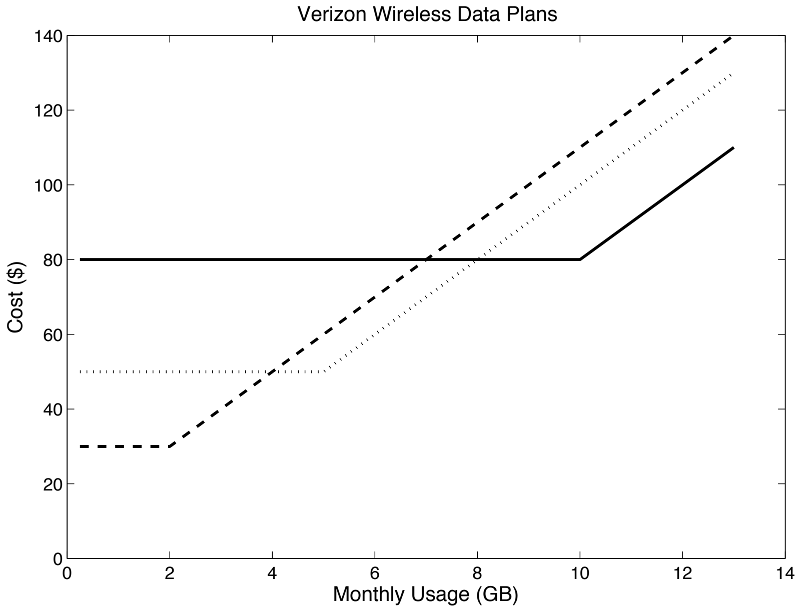


Figure 11.1 Verizon Wireless' data plan options in spring 2012. The plans have a flat-rate component then a usage-based component, e.g., $10 per GB, beyond that.

content consumption is suppressed as a result (which does not have to be the case, as we will see in the next chapter), usage pricing will influence the entire industry ecosystem, including consumers, network providers, content providers, app developers, device manufacturers, and advertisers.

Yet we will see that there are several strong reasons, including those in the interests of consumers, that support usage-based pricing as a better alternative to flat-rate pricing. Whether $10/GB is the right price or not is another matter. We will investigate the pros and cons of usage-based pricing from all these angles.

Despite the different names attached to them, there are two common characteristics of these usage-based pricing plans.

- Charge based on total monthly usage. It does *not* matter when you use it, where you use it, or what you use it for.
- There is a baseline under which the charge is still flat-rate. Then a single straight line with one slope, as the usage grows. The actual numbers in Verizon Wireless cellular data plans in 2012 are shown in Figure 11.1.

11.1.1 Factors behind pricing-plan design

Charging based on consumption probably should have sounded intuitive. That is how most utilities and commodities are charged. But to those who are used to flat-rate Internet connectivity, it represents a radical break. There are two typical precursors to the introduction of usage pricing.

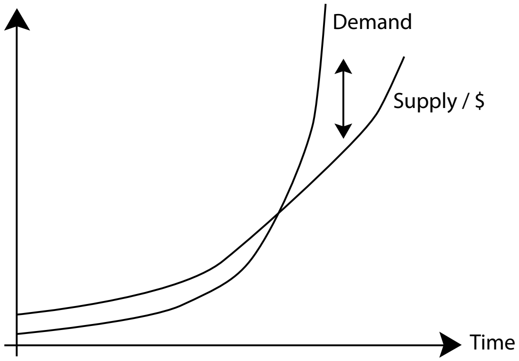


Figure 11.2 The trend of demand and supply/$ of wireless cellular capacity over time. Demand has caught up with supply (per dollar of cost) in recent years, through turning points such as the introduction of iPhone by AT&T in the USA in June 2007, which caused a 50-fold jump in cellular data demand. More importantly, demand is projected to keep growing at a faster pace than supply/$.

- Network usage has surged across many demographics and is projected to climb even higher and faster, e.g., after an ISP introduces iPhones, Android smartphones, and iPads. These devices dramatically enhance the mobile Internet experience and offer many bandwidth-intensive applications and multimedia content. (A more proper word is "capacity-intensive," but we stick to the convention in this field of using the term "bandwidth" in this and the next chapter.) An ISP's profit is the difference between revenue and cost. While demand is rapidly increasing, revenue also needs to catch up with the cost of supporting the rising demand.
- Government regulation allows pricing practices that match cost. There are other regulatory issues that we will discuss soon, but allowing the monthly bill to be proportional to the amount of usage is among the least controversial ones.

So why did the Internet Service Providers (**ISPs**), also called carriers, in countries like the USA avoid usage pricing for many years? There were several reasons, including the following two.

- As the Internet market picked up, each carrier had to fight to capture its market share. A flat-rate scheme is the simplest and easiest one to increase both the overall market acceptance and a particular carrier's market share.
- The growth in the supply of capacity per dollar (of capital and operational expenditure) could still match the growth in demand for capacity.

Then why did the US carriers change to usage pricing during 2010-2012?

- As illustrated in Figure 11.2, the growth rate of demand is outpacing the growth rate of supply/$, and the gap between the two curves is projected

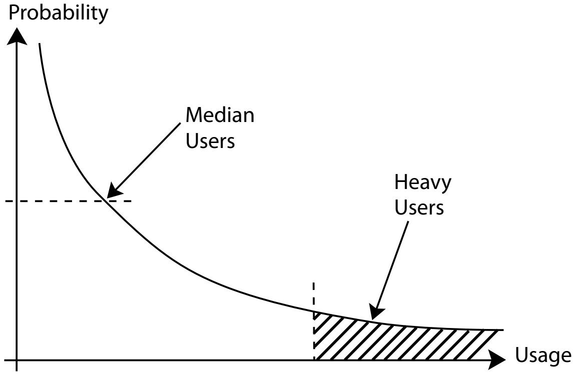


Figure 11.3 Distribution of users' capacity demand, with a long tail. The tail users dictate an ISP's cost structure in both capital expenditure and operational expenditure. If the revenue model is based on the median user, the mismatch between cost and revenue will grow as the tail becomes longer.

to widen even further in the coming years. This we call the "Jobs' inequality of capacity." Once the device suppliers and application communities, such as Apple and iOS app developers, figured out how to make it easy and attractive for users to consume mobile Internet capacity, innovation in those spaces proceeded faster than the supply side can keep up with. Cisco predicts that the mobile Internet demand will keep doubling every year. That is more than 64 times after five years. No technology can double the supply/$ each year forever.

- If we look at the distribution of capacity demand, the tail of that distribution, shown in Figure 11.3, is often the dominant factor in an ISP's cost structure. That tail has always been long, but is getting longer and longer now. If the ISP still collects revenue based on the median user, the difference between cost and revenue will be too big.

One way or another, the cost of building and operating a network must be paid by someone. Usage pricing based on monthly consumption, however, is not the only way to tackle the above issues. ISPs have other choices, such as the following.

- Increase the flat rate for everyone as demand increases. With a sufficiently high flat rate, the revenue collected will be adequate. But clearly this creates affordability and fairness issues.
- Cap heavy users' traffic. Once you exceed a cap, you can no longer use the network. Or the speed will be throttled to the point that the quality of service becomes too low for practical use once the cap has been exceeded. This is actually a special case of usage pricing: the pricing slope becomes infinite beyond the baseline.
- Slow down certain classes of traffic. For example, for a period of time, Comcast throttled BitTorrent users, who often had massive amounts of file sharing

and movie downloads using the popular P2P service. This may raise concerns on network neutrality.

- Offload some of the cellular traffic to open and non-metered WiFi networks operating in unlicensed frequency bands. But as we will see in Chapters 18 and 19, mobility support, intereference management, coverage holes, and backhaul capacity limitation can all become bottlenecks to this solution.
- Implement smarter versions of usage pricing, as discussed in the next chapter.

Most of the ISPs have realized the problem and started pursuing the usage-pricing solution. What are the criteria that we can use to compare alternative solutions to the exploding demand for mobile data? There are too many to list here, but the top ones include the following.

- *Economic viability*: As profit-seeking companies, ISPs need to first recover cost and then maximize profit. Their profit margins are in general declining, as many other transportation businesses have seen in the past. Mobile and wireless networks are bright spots that they need to seize.
- *Fairness*: Consumer A should not have to use her money to subsidize the lifestyle of consumer B.
- *Consumer choice*: Consumers should be able to choose among alternatives, e.g., spend more money to get premium services, or receive standard services with a cost saving.

Along all of the above lines, usage pricing makes more sense than fixed pricing, although it can be further enhanced with more intelligence as we will describe in the next chapter. The key advantages of usage pricing are listed below and will be analyzed in detail in the next section.

- Usage pricing produces less “waste” and matches cost.
- Usage pricing does not force light users to subsidize heavy users.
- Usage pricing helps with better differentiation in the quality of using the Internet.

11.1.2 Network neutrality debates

Before we move to the next section, it is worthwhile to mention “network neutrality,” a central policy debate especially in the USA. Counter-productive to useful dialogues, this “hot” phrase has very different meanings to different people. Usually there are three layers of meanings.

- *Access/choice*: Consumers should have access to all the services offered over the Internet, and a choice of how they consume capacity on the Internet.
- *Competition/no monopoly*: ISPs should have no monopoly power and the marketplace needs to have sufficient competition.
- *Equality/no discrimination*: All traffic and all users should be treated the same. This may actually contradict the requirement of access/choice.

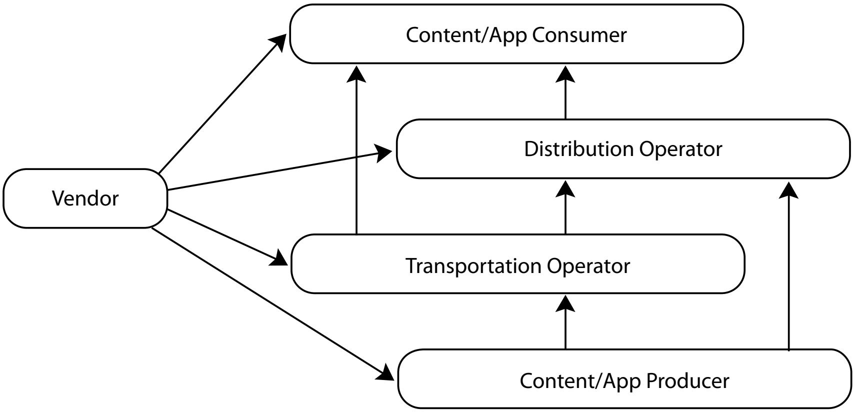


Figure 11.4 Five-party interactions in the industry. Content/app producers include YouTube and Deja, transportation operators include AT&T and Comcast, distribution operators include Akamai and BitTorrent. "Shortcuts" have been created in the traditional food chain, from content/app producers directly to distribution operators, and from transportation operators directly to consumers, further complicating the industry interaction. There is also often a lack of information visibility or incentives for higher efficiency across the boundaries of these parties.

While the last point might sound like an ideal target, it is sometimes neither feasible nor helpful to carry it out. There are four types of "no discrimination," depending on what "discrimination" means.

- *Service limitation*: Because of vertical integration, an ISP also becomes a content owner, possibly blocking access to other content. Or an ISP could block access to voice-call apps on iPhones in order to generate more revenue for its own voice business.
- *Protocol-based discrimination*: Certain protocols generate a significant amount of heavy traffic, e.g., BitTorrent, and get blocked.
- *Differentiation of consumer behaviors*: Usage pricing is one of the simplest ways to correlate pricing with consumer behavior; if consumer A takes up more capacity, she pays more.
- *Traffic management and quality-of-service provisioning*: Examples include maintaining more than one queue in a router, scheduling traffic with weighted fair queuing, or prioritizing emergency traffic like healthcare-monitor signals over non-essential software updates. (We will discuss some of these quality-of-service mechanisms in Chapter 17.)

While neutrality against service limitations is essential, neutrality against protocol-discrimination is debatable, neutrality against consumer behavior differentiation is harmful, and neutrality against traffic management is downright impossible: if having more than one queue is anti-neutral, then the Internet has never been and never will be neutral.

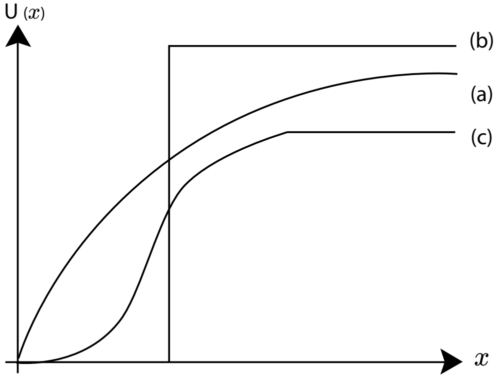


Figure 11.5 Three examples of utility-function shapes: (a) concave, (b) discontinuous, and (c) sigmoidal. Eventually utility functions all become concave as marginal returns diminish. Maximizing concave and smooth utility functions is mathematically easier than maximizing sigmoidal or discontinuous utility functions.

In fact, a naive view of "equality" harms the tenet of providing access and choices to consumers, often viewed as a more important component of neutrality. As summarized by the Canadian Radio, Television and Communications office: "Economic practices are the most transparent Internet traffic management practices," and we should "match consumer usage with willingness to pay, thus putting users in control and allowing market forces to work."

There is much more to the network-neutrality debate than we have space for in this chapter. This debate is further complicated by the fairness and efficiency issues arising out of the five-party interactions shown in Figure 11.4. We will now turn to some basic modeling language about these interactions.

11.2 A Long Answer

11.2.1 Utility maximization model

In order to proceed further to understand Internet access pricing, we need to build some model of consumer demand. The **utility function** is a common modeling tool in economics to capture "how happy" a user would be if a certain amount of resource is allocated. In Chapters 1 and 2, we saw payoff functions in games. Utility functions are a generalization of these. They further lead to models of strategic thinking by users, which are based on assumptions in the expected utility theory and its many extensions.

A typical utility function is shown as curve (a), a logarithmic function, in Figure 11.5. We denote the utility function of session i as $U_i(x_i)$, where x_i is some performance metric like throughput. Maximizing the *sum* of utilities across all users, $\sum_i U_i(x_i)$, is referred to as **social welfare maximization**. It is that

12 How can I pay less for each GB?

12.1 A Short Answer

ISPs charging consumers on the basis of usage is just one corner of the overall landscape of Internet economics. We will pick consumers' monthly bills to focus on in this chapter, but there are many other key questions.

- The formation of the Internet is driven in part by economic considerations. Different ISPs form peering and transit relationships that are based on business and political decisions as much as on technical ones.
- The invention, adoption, and failure of Internet technologies are driven by the economics of vendor competition and consumer adoption.
- The investment of network infrastructure, from purchasing wireless licensed spectrum to deploying triple-play broadband access, is driven by the economics of capital expenditure, operational expenditure, and returns on investment.

The economics of the Internet are interesting because the technology–economics interactions are *bidirectional*: economic forces shape the evolution of technology, while disruptive technologies can rewrite the balance of economic equations. This field is also challenging to study because of the lack of publicly available data on ISPs' cost structures and the difficulty of collecting well-calibrated consumer data.

12.1.1 Smart data pricing

There is a rapidly growing research field and industry practice on network access pricing. What we described on usage pricing in the last chapter, in the form of tiered and then metered/throttled plans, is just a starter. A few possibilities of **Smart Data Pricing** (SDP) are listed below.

- The hourly-rate model, e.g., Mobinil in Egypt charges data connection by the number of hours of usage.
- Expected-capacity pricing, which relies on resource allocation driven by the needs of different sessions rather than just the byte-counts. Characterizing a session's needs, however, can be tricky, even after a period of performance observation during trials.

- Priority pricing, where you can pay more to get a higher speed, such as the priority pass service by SingTel in Singapore. A turbo mode of anti-throttling is also being considered in the USA for heavy users whose speed is throttled once usage exceeds some threshold. In an elaborate form, priority pricing may even take the form of an auction where the price reflects the negative externality imposed on other users by boosting your speed. Paris metro pricing adds another interesting variation.
- Two-sided pricing, where an ISP charges either the content consumers or the content producers, or both. It is used by Telus in Canada and TDC in Denmark. This can also become an application-dependent pricing method.
- Location-dependent pricing, which is also used in the transportation industry in certain cities, e.g., downtown London and Singapore.
- Time-dependent pricing, which is also used in certain utility industries e.g., energy networks, and will be elaborated in this chapter.

In static pricing, time periods and the associated prices are predetermined and do not vary except over very long timescales like months and years. In dynamic pricing, network access prices are continuously adjusted to reflect the state of the network. We will see that congestion control in Chapter 14 can be interpreted as a type of dynamic pricing.

In this chapter, we bring up several topics that illustrate some central themes in the field. One is charging that is based on *when* the Internet is used, and the other is *differentiating service qualities* by simply charging different prices. We will also explore the question of "whom to charge" through two-sided pricing. These are some of the possibilities to help the entire network ecosystem, from consumers to ISPs, and from content providers to advertisers, move from the shadow of \$10/GB to win-win solutions. In a win-win,

- ISPs generate more revenue, lowers cost, and reduces churn;
- consumers pay less per GB of data;
- content providers attract more eyeballs; and
- vendors sell more innovative software and hardware.

12.1.2 Time-dependent pricing

Pricing based just on monthly bandwidth usage still leaves a timescale mismatch: ISP revenue is based on monthly usage, but peak-hour congestion dominates its cost structure. Ideally, ISPs would like bandwidth consumption to be spread evenly over all the hours of a day. **Time-Dependent Pricing** (TDP) charges a user according to not just "how much" bandwidth is consumed but also "when" it is consumed, as opposed to Time-Independent usage Pricing (TIP), which considers only monthly consumption amounts. For example, the day-time (counted as part of minutes used) and evening-weekend-time (free) differentiation, long practiced by wireless operators for cellular voice services, is a simple two-period TDP scheme. Multimedia downloads, file sharing, social media updates, data backup, and software downloads, and even some streaming applications all have various degrees of time elasticity.

As an idea as old as the cyclic patterns of peak and off-peak demand, TDP has been used in transportation and energy industries. Now it has the potential to even out time-of-day fluctuations in (mobile) data consumption: when data plans were unlimited, \$1 a GB was infinitely expensive, but now with \$10 a GB becoming the norm, \$8 a GB suddenly looks like a bargain. As a pricing practice that does not differentiate in terms of traffic type, protocol, or user class, it also sits lower on the radar screen of the network neutrality debate. TDP time-multiplexes traffic demands. It is a counterpart to spatial multiplexing in Chapter 13 and to frequency multiplexing in Chapter 18.

Much of the pricing innovation in recent years has occurred outside the USA. Network operators in highly competitive markets, e.g., in India and Africa, have adopted innovative dynamic pricing for voice calls.

- The African operator, MTN, started "dynamic tariffing," a congestion-based pricing scheme in which the cost of a call is adjusted every hour in each network cell depending on the level of usage. Using this pricing scheme, instead of a large peak demand around 8 pm, MTN Uganda found that many of its customers were waiting to take advantage of cheaper call rates.
- A similar congestion-dependent pricing scheme for voice calls was also launched in India by Uninor. It offers discounts to its customers' calls that depend on the network traffic condition in the location of the call's initiation.
- Orange has been offering "happy hours" data plans during the hours of 8–9am, 12–1pm, 4–5pm, and 10–11pm.

We have to face two questions here. Can we effectively parameterize **delay sensitivity** in setting the right prices? Are users willing to defer their Internet traffic in exchange for a reduced monthly bill? Ultimately, it is the ratio between demand elasticity and delay sensitivity (for each user and each application) that determines how much can time-dependent pricing help.

12.2 A Long Answer

12.2.1 Thinking about TDP

Usage-based pricing schemes use penalties to limit network congestion by reducing demand from heavy users. However, they cannot prevent the peak demand by many users from concentrating during the same time periods. ISPs must provision their network in proportion to these peak demands, leading to a timescale mismatch: ISP revenue is based on monthly usage, but peak-hour congestion dominates its cost structure. Empirical usage data from typical ISPs shows large fluctuations even on the timescale of a few minutes. Thus, usage can be signif icantly evened out if a TDP induces users to time-shift their demand. However, a simple two-period, time-dependent pricing scheme (e.g., different prices for

13 How does traffic get through the Internet?

We have mentioned the Internet many times so far, and all the previous chapters rely on its existence. It is about time to get into the architecture of the Internet, starting with these two chapters on the TCP/IP foundation of the Internet.

13.1 A Short Answer

We will be walking through several core concepts behind the evolution of the Internet, providing the foundation for the next four chapters. So the "short answer" section is going to be longer than the "long answer" section in this chapter.

It is tricky to discuss the historical evolution of technologies like the Internet. Some of what we would like to believe to be the inevitable results from careful design are actually the historical legacy of accidents, or the messy requirements of backward compatibility, incremental deployability, and economic incentives. It is therefore not easy to argue about what could have happened, what could have been alternative paths in the evolution, and what different tradeoffs might have been generated.

13.1.1 Packet switching

The answer to this chapter's question starts with a fundamental idea in designing a network: when your typical users do not really require a *dedicated* resource, you should allow users to *share* resources. The word "user" here is used interchangeably with "session." The logical unit is an application session rather than a physical user or device. For now, assume a session has just one source and one destination, i.e., a **unicast** session.

In the case of routing, the resource lies along an entire path from one end of a communication session, the sender, to the other end, the receiver. We can either dedicate a fixed portion of the resources along the path to each session, or we can mix and match packets from different sessions and also share all the paths. This is the difference between **circuit-switched** and **packet-switched** networks.

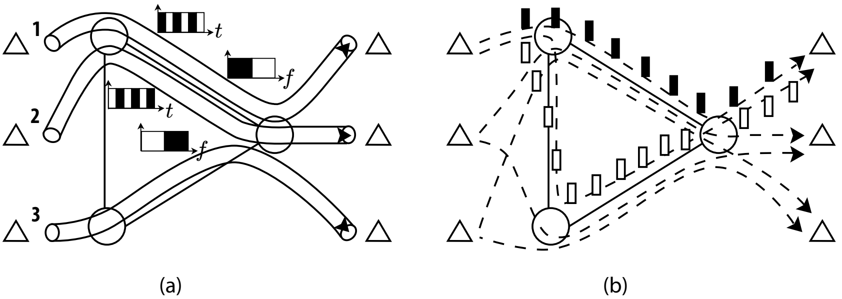


Figure 13.1 A simple network with three interconnected routers and three sessions. (a) Circuit switching: each session gets a dedicated circuit, either a portion of each timeslot t or a fraction of the frequency band f, even when it is not used. (b) Packet switching: each session sends packets along one or more paths (when there are packets to send) and all paths are shared across timeslots and frequency bands.

Before the 1960s, networking was mostly about connecting phone calls in circuit-switched Public Switched Telephone Networks (**PSTNs**). There continued to be active research all the way to the early 2000s, including dynamic routing as you will see in a homework problem.

A revolution, which came to be known as the Internet, started during the 1960s–1970s, witha shift to packet switching as the fundamental paradigm of networking. In the early 1960s, researchers formally developed the idea of chopping up a session's messages into small packets, and sending them along possibly different paths, with each path shared by other sessions. Figure 13.1 contrasts circuit switching with packet switching. Each circuit in circuit switching may occupy either a particular frequency band or a dedicated portion of timeslots. In contrast, in packet switching, there is no dedicated circuit for each session. All sessions have their packets sharing the paths.

In 1969, sponsored by the US Advanced Research Project Agency (ARPA), UCLA and three other institutions put together the first prototype of a packet-switched network, which came to be known as the **ARPANET**. The ARPANET started to grow. In 1974, Cerf and Kahn developed a protocol, i.e., a set of rules for communication among the devices, for packet-switched networks, called the Transmission Control Protocol/Internet Protocol (**TCP/IP**). This protocol enabled scalable connectivity in the ARPANET. From 1985 to 1995, the US National Science Foundation (NSF) took over the next phase of development, sponsoring the creation and operation of an ever-increasing *network of networks* called the **NSFNET**. Starting in the early 1990s, commercial interests and entrepreneurial activities dramatically expanded this inter-connected network of networks. Indeed, by 1994, the World Wide Web and web browser user-interface had matured, and the world quickly moved into commercial applications built on top of this network, known by then as the Internet. Today the Internet has blossomed into an essential part of how people live, work, play, talk, and think.

There are now more Internet-connected devices than people in the world, and it is projected that by 2020 there will be six times as many connected devices as people. It has been a truly amazing five decades of technology development.

The debate between dedicated resource allocation and shared resource allocation runs far and deep. In addition to circuit vs. packet switching here and orthogonal vs. non-orthogonal resource allocation in Chapter 1, we will also see three more special cases of this design choice: client-server vs. peer-to-peer, local storage vs. cloud services, and contention-free scheduling vs. random access in Chapters 15, 16, and 18, respectively.

There is one big advantage of circuit switching, or dedicated resource allocation in general: *guarantee of quality.* As each session gets a circuit devoted to it, throughput and delay performance are accordingly guaranteed, and there is very little jitter (the variance of delay). In contrast, in packet switching, a session's traffic is (possibly) split across different paths, each of which is shared with other sessions. Packets arrive out of order and need to be re-ordered at the receiver. Links may get congested. Throughput and delay performance become uncertain. Internet researchers call this the **best-effort** service that the Internet offers, which is perhaps more accurately described as *no effort* to guarantee performance.

On the other hand, there are two big advantages of packet switching: (1) *ease of connectivity* and (2) *scalability due to efficiency.*

(1) Ease of connectivity is easy to see: there is no need to search for, establish, maintain, and eventually tear down an end-to-end circuit for each session.

(2) Scalability here refers to the ability to take on many diverse types of sessions, some long-duration ones, others short bursts, and to take on many of them. There are two underlying reasons for the efficiency of packet switching, which in turn leads to high scalability. These two reasons correspond to the "many sessions share a path" feature and the "each session can use multiple paths" feature of packet switching, respectively. We call these two features statistical multiplexing and resource pooling.

- **Statistical multiplexing**: packet switching can flexibly map demand of capacity onto supply of capacity. This suits the dynamic, on-demand scenarios with bursty traffic. In particular, when a source is idle and not sending any traffic onto the network, it does not occupy any resources.
- **Resource pooling**: this one takes a little math to demonstrate, as we will in a homework problem. But the basic idea is straightforward: instead of having two sets of resources (e.g., two links' capacities) in isolation, putting them into a single pool lowers the chance that some demand must be turned down because one set of resources is fully utilized.

In the end, the abilities to easily provide connectivity and to scale up with many diverse users won the day, although that was not clear until the early 2000s. In contrast to quality guarantee, which is certainly nice to have, these properties are *essential* to have for a dynamic and large network like the Internet. Once the

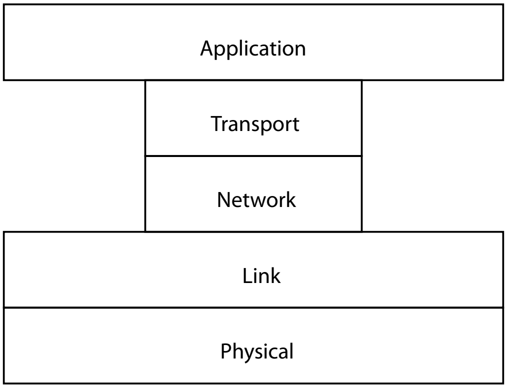


Figure 13.2 Modularization in networking: A typical model of a layered protocol stack. Each layer is in charge of a particular set of tasks, using the service provided by the layer from below and in turn providing a service to the layer above. The horizontal lines that separate the layers represent some kind of limitation of what each layer can see and can do. Over the years, the applications have evolved from file transfer based on command-line inputs to all those applications we experience today. The physical and link layer technologies have evolved from 32 kbps dial-up modem to 10 Gbps optic fibers and 100 Mbps WiFi. The two middle layers, however, dominated by TCP/IP, have remained largely unchanged over the years. They are the "thin waist" of the "hour-glass" model of the protocol stack.

network has grown in an easy and scalable way, we can search for other solutions to take care of quality variation. But you have to grow the network *first*, in terms of the number of users and the types of applications. This is a key reason why IP took over the networking industry and packet switching prevailed, despite alternative designs in protocols (that we will not cover here) like X.25, ATM, frame relay, ISDN, etc.

13.1.2 Layered architecture

Managing a packet-switched network is complicated. There are many tasks involved, and each task's control requires a sequence of communication and computation called a **protocol** to control it. It is a natural practice when engineering such a complex system to break it down into smaller pieces. This process of **modularization** created the **layered protocol stack** for the Internet. The idea of modularizing the design is *not* motivated by efficiency of resource allocation, but by economic viability through the business models of different companies specializing in different layers, and by the robustness regarding unforeseen innovations that may ride on the Internet. This evolvability is further enhanced by the overlay networks that can create new network topologies and functionalities on top of the Internet connectivity, as we will see in Chapter 15.

A typical layered protocol stack is shown in Figure 13.2. TCP/IP sits right in the middle of it in the transport and network layers. Over the short span of Internet evolution, the physical medium's transmission speed has gone up more than 30,000 times, and the applications have gone from command-line-based file transfer to Netflix and Twitter. Yet the Internet itself continued to work, thanks in large part to the "thin waist" of TCP/IP that stayed mostly the same as the applications and the communication media kept changing.

Each layer provides a service to the layer above, and uses a service from the layer below. For example, the transport layer provides an end-to-end connection, running the services of session establishment, packet re-ordering, and congestion control, to the application layer above it that runs applications such as the web, email, and content sharing. In turn, the transport layer takes the service from the network layer below it, including the connectivities established through routing. The link layer is charged with controlling the access to the communication medium, and the physical layer controls the actual transmission of information on the physical medium.

There are functional *overlaps* across layers. For example, the functionality of error control is allocated to many layers: there is error control coding in the physical layer, hop-by-hop retransmission at the link layer, multipath routing for reliability in the network layer, and end-to-end error checking at the transport layer. Functional redundancy is not a bug, it is there by design, paying the price of efficiency reduction for robustness and clear boundaries among the layers.

How should we allocate functionalities among the layers and put them back together at the right interface and timescale? That is the question of **network architecture** that we will continue to explore in later chapters. For example, the horizontal lines in Figure 13.2, denoting the boundaries between protocol layers, are actually very complicated objects. They represent limitations as to what each layer can *do* and can *see*. In the next chapter, we will get a glimpse of some methodologies to understand this architectural decision of "who does what" and "how to glue the modules together."

Just between the transport and network layers, there are already quite a few interesting architectural decisions made in TCP/IP, the dominant special case of the layers 4/3 protocol. First, the transport layer, in charge of end-to-end management, is *connection-oriented* in TCP, whereas the network layer, in charge of connectivity management, runs hop-by-hop *connectionless* routing in IP. As an analogy, calling someone on the phone requires a connection-oriented session to be established first between the caller and the callee. In contrast, sending mail to someone needs only a connectionless session since the recipient does not need to know there is a session coming in. The design choice of connection-oriented TCP and connectionless IP follows the "end-to-end" principle that end-hosts are intelligent and the network is "dumb." Connectivity establishment should be entirely packet switched in the network layer, and end-to-end feedback run by the layer above. But this design choice was *not* the only one that the Internet

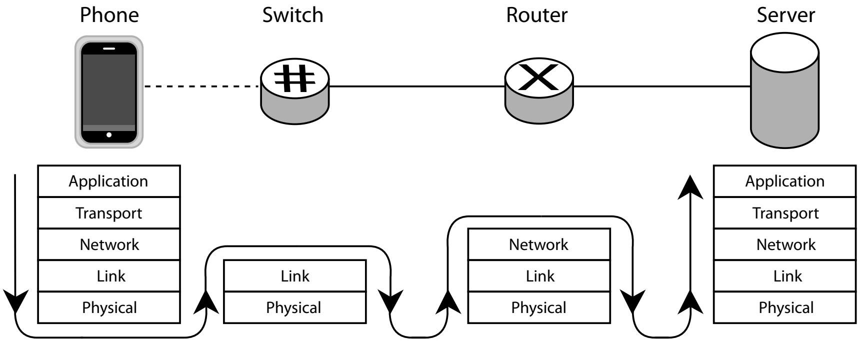


Figure 13.3 Different network elements process up to different layers in the protocol stack. The end-hosts process all the way up to the application layer. Switches that forward frames process up to the link layer. Routers that move datagrams across the network process up to the network layer. In a realistic end-to-end connection, there are usually many more hops.

tried over its decades of evolution, e.g., a connectionless transport layer on top of a connection-oriented network layer is also possible and indeed was once used.

Second, routing in IP is independent of load conditions on the links, whereas congestion control in TCP takes care of managing demand at the end-hosts in response to link loads. In addition to the end-to-end principle, this strategy assumes that rate adaptation at the end hosts is easier to stabilize than route adaptation inside the network.

As we will see in a homework problem, there is also an interesting architectural division-of-labor between the network layer and the link layer below it.

Zooming out of the protocol stack again, the application layer runs applications that generate a sequence of **messages**. Each of these is divided into **segments** at the transport layer, with a layer 4 **header** added in front of the actual content, called **payload**. Then it is passed on to the network layer, which divides and encapsulates the segments as datagrams or **packets**, with a layer 3 header in the front. Each datagram is further passed on to the link layer, which adds another layer 2 header to form a **frame**. This is finally passed on to the physical layer for transmission. These headers are overheads, but they contain useful, sometimes essential, identification and control information. For example, the layer 3 header contains the source node's address and the destination node's address, which are no doubt useful to have in routing. We will examine the impact of these semantic overheads on performance in Chapter 19.

Each network element, e.g., your home gateway, your company's WiFi controller, the central office equipment near the town center, the big router inside the "cloud," runs a subset of the layered protocol stack. Each will decode and read the header information associated with its subset. This is illustrated in Figure 13.3.

13.1.3 Distributed hierarchy

The Internet is not just complex in terms of the number of tasks it has to manage, but also big in terms of the number of users. Hierarchy becomes essential. An end-to-end session, for example, a YouTube streaming session from Google servers to your iPhone may traverse a wireless air-interface, a few links in the cellular core network, and then a sequence of even more links across possibly multiple ISPs in the public Internet.

While *modularization* helps take care of the complexity by "divide and conquer" in terms of functionalities, *hierarchy* helps take care of the large size by "divide and conquer" in terms of the physical span. This is a recurring theme in many chapters. In the current one, we see that the Internet, this network of networks with more than 30,000 **Autonomous Systems** (ASs), has several main hierarchical levels as illustrated in Figure 13.4.

- A few very large ISPs with global footprints are called **tier-1 ISPs**, and they form a *full-mesh* **peering** relationship among themselves: each tier-1 ISP has some connections with each of the other tier-1 ISPs. This full mesh network is sometimes called the Internet backbone. Examples of tier-1 ISPs include AT&T, BT, Level 3, Sprint, etc.
- There are many more tier-2 ISPs with regional footprints. Each tier-1 ISP is connected to some tier-2 ISPs, forming a **customer–provider** relationship. Each of these tier-2 ISPs provides connectivity to many tier-3 ISPs, and this hierarchy continues. The point at which any two ISPs are connected is called the Point of Presence (PoP).
- An ISP of any tier could be providing Internet connectivity directly to consumers. Those ISPs that take traffic only to or from their consumers, but not any transit traffic from other ISPs, are called stub ISPs. Typically, campus, corporate, and rural residential ISPs belong to this group.

Another useful concept in distributed hierarchy is that of a **domain**. Each business entity forms a domain called an AS. There is often a centralized controller within each AS. As we will see later in this chapter, routing *within* an AS and routing *across* ASs follow very different approaches.

Later, in Chapter 15, we will also see how functional and spatial hierarchies combine in building overlay networks.

13.1.4 IP routing

Packet switching, layered architecture, and distributed hierarchy are three fundamental concepts of the Internet. With those topics discussed, we can move on to routing in the Internet.

Transportation networks often offer interesting analogies for communication and social networks. In this case, we can draw a useful analogy from the postal

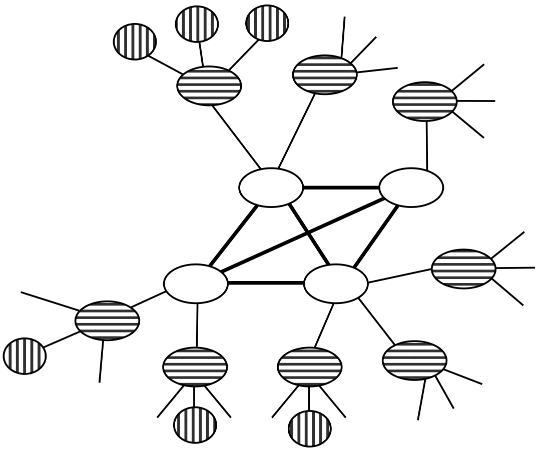


Figure 13.4 Spatial hierarchy in networking: Multiple levels of ISPs and their relationships. Each node in this graph is an ISP, and each link represents a business relationship and physical connections between two ISPs. The four ISPs in the center are tier-1 ISPs, with peering links among themselves. Each of them provides connectivity to many customer ISPs. The stub ISPs at the edge do not provide transit service to any other ISPs. An ISP at any tier may also provide connections to the end-users, which are not shown here.

mail service. In order to route a letter from a sender to the receiver, we need three main functionalities.

- *Addressing.* We first need to attach a unique label to each node in the network, for otherwise we cannot even identify sources and destinations. In the mail system, the label is the postal address, like a street address or mailbox number. Zip codes can quickly zoom you into a subnetwork of the country. In the Internet, we use the **IP address**, a 32-bit number often represented as four decimal numbers separated by dots. Each of these four numbers ranges from 0 to 255 since it is specified by 32/4=8 bits, for example, 127.12.5.88. "Zip codes" here are called subnet masks, for example, 127.12.5.0/24 means that the first 24 bits give the **prefix** of all this subnet's IP addresses: each IP address in this subnet must start with 127.12.5, and can end with any 8 bits. However, in the mail system, an address and a person's ID are separated. In the Internet, an IP address is both an address for establishing connectivity and an identifier of a device. This double loading of functionality onto IP addresses caused various control problems in the Internet.
- *Routing.* Then you have to decide the paths, either one path for each session (single-path routing) or multiple paths for each session (multipath routing). Postal mail uses single-path routing, and routing decides ahead of time which intermediate cities the mail goes through in order to reach, say,

Princeton, NJ, from Stanford, CA. There are two broad classes of routing methods: *metric*-based and *policy*-based routing. Inside an AS, routing is based on some kind of metric, either picking the shortest path between the given source and destination, or distributing the traffic across the paths so that no single path is too loaded. In between the ASs, however, routing is based on policies. For example, AS 1 might suspect there are hackers connected through AS 2, therefore it avoids routing packets along any path traversing AS 2.

- *Forwarding.* Forwarding implements the routing policy. The actual action of forwarding happens each time a packet is received at a router, or each letter is received at an intermediate post office. Some forwarding mechanisms look only at the destination address to decide the next hop, while others, like MultiProtocol Label Switching (**MPLS**), read some labels attached to the packet that explicitly indicate the next hop. In any case, a forwarding decision is made, and one of the egress links connected to the router is picked to send the packet.

Let us look at each of the above in a little more detail now, before focusing the rest of the chapter on just the routing portion.

There are two versions of IP: version 4 and version 6. IPv4 uses 32 bits for addresses, which ran out as of early 2011. IPv6 uses four times as many bits, 128 bits, translating into 2^{128}, about 10^{39}, available addresses. That might sound like a lot, but with the proliferation of Internet-connected devices, we are well on our way to using many of these addresses. One way to upgrade an IPv4 network into IPv6 is to create a "tunnel" between two legacy IPv4 network elements, where IPv6 packets are encapsulated in IPv4 headers.

How are these IP addresses allocated? They used to be given out in blocks, with different block sizes determined by the "class". For example, each class-A address block has a fixed 8-bit prefix, so $2^{32-8} = 2^{24}$ addresses in a class-A block. That is usually given to a national ISP or a large equipment vendor. Lower classes have fewer addresses per block. But this coarse granularity of 8-bit blocks introduced a lot of waste in allocated but unused IP addresses. So the Internet community shifted to Classless InterDomain Routing (**CIDR**), where the granularity does not have to be in multiples of 8 bits.

As a device, you either have a fixed, static IP address assigned to you, or you have to get one dynamically assigned to you by a controller sitting inside the operator of the local network. This controller is called the Dynamic Host Configuration Protocol (**DHCP**) server. A device contacts the DHCP server, receives a currently unused IP address, and returns it to the IP address pool when no longer needed. You may wonder how a device can communicate with a DHCP server in the first place. We will address the protocols involved in Chapter 19. Sometimes the address given to a device within a local network, e.g., a corporate intranet, is different from the one seen by the outside world, and a Network Address Translation (**NAT**) router translates back and forth.

As mentioned, inter-AS routing is very different from intra-AS routing. Border Gateway Protocol (**BGP**) is the dominant protocol for address discovery and reachability for inter-AS routing. It "glues" the Internet together. However, as a policy-based routing protocol, it is a complicated, messy protocol, with many gray areas. We will only briefly describe it in the Advanced Material.

Within an AS, there are two main flavors of metric-based routing protocols: Routing Information Protocol (**RIP**) uses the **distance vector** method, where each node collects information about the distances between itself and other nodes, and Open Shortest Path First (**OSPF**) uses the **linked state** method, where each node tries to construct a global view of the entire network topology. We will focus on the simpler RIP in the next section, saving OSPF for the Advanced Material.

A packet arrives at a router interface, that interface acts as the input port for the packet while another interface acts as the output port. In-between is the switching fabric that physically moves the packet from an input port to the output port. If packets arrive too fast, congestion occurs inside the router. Sometimes it occurs because the intended output ports are occupied, sometimes because the switching fabric is busy, and sometimes because a packet is waiting for its turn at the input port's queue, thus blocking all the packets behind it in the same input queue.

Which output port is the "right" one? That is decided by looking up the forwarding table, which is either stored centrally in the router, or duplicated with one copy at each input port. The forwarding table connects the routing decisions to actual forwarding actions. A common type of forwarding table lists all the destination IP addresses in the Internet, and indicates which output port, thus the next hop router, a packet should go to on the basis of its destination address written in the header. There are too many IP addresses out there, so the forwarding table often groups many addresses into one equivalent class of input.

We are now going to study one member of the intra-AS routing family, and then how forwarding tables are constructed from distributed messages passing among the routers.

13.2 A Long Answer

Consider a directed graph $G = (V, E)$ representing the topology inside an AS, where each node in the node set V is a router, and each link in the link set E is a physical connection from one router i to another router j.

Each link has a cost c_{ij}. It is often a number approximately proportional to the length of the link. If it is 1 for all the links, then minimizing the cost along a path is the same as minimizing the hop count. If it were dynamically reflecting the congestion condition on that link, it would lead to dynamic, load-sensitive routing. But IP does not practice dynamic routing, leaving load sensitivity to TCP congestion control.

14 Why doesn't the Internet collapse under congestion?

14.1 A Short Answer

14.1.1 Principles of distributed congestion control

When demand exceeds supply, we have congestion. If the supply is fixed, we must reduce the demand to alleviate congestion. Suppose the demand comes from different nodes in a network, we need to coordinate it in a distributed way.

As the demand for capacity in the Internet exceeds the supply every now and then, **congestion control** becomes essential. The timescale of congestion control is on the order of ms, in contrast to shaping consumer behavior through pricing in Chapters 11 and 12. The need for congestion control was realized in October 1986, when the Internet had its first congestion collapse. It took place over a short, three-hop connection between Lawrence Berkeley Lab and UC Berkeley. The normal throughput was 32 kbps (that is right, kbps, not the Mbps numbers we hear these days). That kind of dial-up modem speed was already low enough, but during the congestion event, it dropped all the way down to 40 bps, by almost a factor of 1000.

The main reason was clear as we saw from the last chapter on routing: when users send so many bits per second that their collective load on a link exceeds the capacity of that link, these packets are stored in a buffer and they wait in the queue to be transmitted. But when that wait becomes too long, more incoming packets accumulate in the buffer until the buffer overflows and packets get dropped. This is illustrated in Figure 14.1.

These dropped packets never reach the destination, so the intended receiver never sends an acknowledgement (an ACK packet) back to the sender, as it should do in the **connection-oriented**, end-to-end control in TCP. As mentioned in the last chapter, Internet design evolution considered different divisions of labor between layers 3 and 4, eventually settling on a connection-oriented layer 4 and connectionless layer 3 as the standard configuration. According to TCP, the sender needs to resend the unacknowledged packets. This leads to a vicious cycle, a *positive-feedback loop* that feeds on itself: congestion persists as the same set of senders that caused congestion in the first place keeps resending the dropped packets. Packets keep getting dropped at the congested link, resent from the source, dropped at the congestion link ... Senders need to rethink how they can avoid congestion in the first place, and they need to back off when congestion

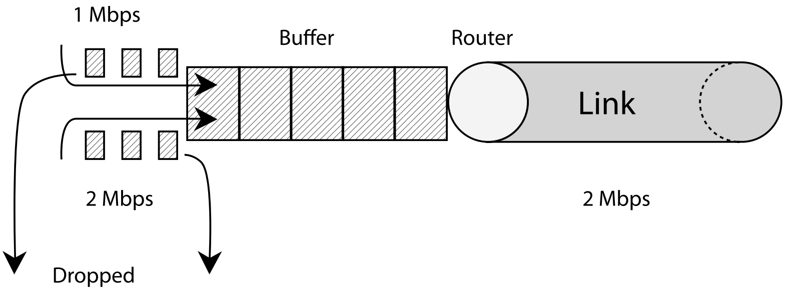


Figure 14.1 An illustration of congestion at one end of a link. Two sessions arrive at the buffer with an aggregate demand of 3 Mbps, but there is only a supply of 2 Mbps in the outgoing link. The buffer is filled up and packets start to get dropped. Which packets get dropped depends on the details of the queue-management protocols.

happens. We need to turn the positive-feedback loop into a *negative*-feedback loop.

That was what Van Jacobson proposed in the first congestion control mechanism added to TCP in 1988, called **TCP Tahoe**. It has been studied extensively since then, and improved significantly several times. But most of the essential ideas in congestion control for the Internet were in TCP Tahoe already.

- *End-to-end control via negative feedback.* We can imagine congestion control within the network where, hop by hop, routers decide for the end hosts at what rates they should send packets. That is actually what another protocol, called Asynchronous Transmission Mode (**ATM**), does to one type of its traffic, the Arbitrary Bit Rate traffic. But TCP congestion control adopts the alternative approach of having an *intelligent edge network* and a *dumb core network*. The rate at which a sender sends packets is decided by the sender itself. But the network provides hints through some feedback information to the senders. Such feedback information can be inferred from the *presence* and *timing* of acknowledgement packets, transmitted from the receiver back to the sender acknowledging the in-order receipt of each packet.
- *Sliding-window-based control.* If a sender must wait for the acknowledgement of a sent packet before it is allowed to send another packet, it can be quite slow. So we pipeline by providing a bigger allowance. Each sender maintains a sliding window called the **congestion window**, with its value denoted by `cwnd`. If the window size is 5, that means up to five packets can be sent before the sender has to pause and wait for acknowledgement packets to come back from the receiver. For each new acknowledgement packet received by the sender, the window is slided one packet forward and this enables the sending of a new packet, hence the name "sliding window." This way of implementing a restriction on transmission rate introduces the

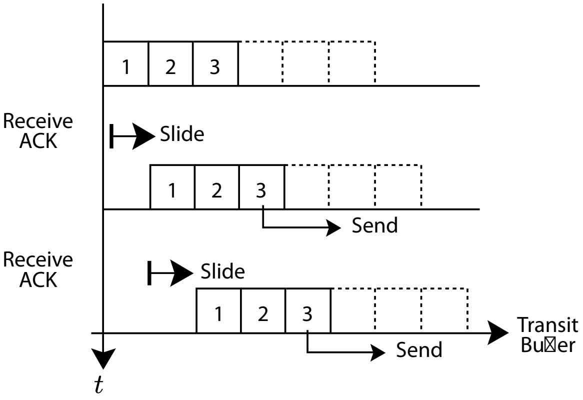


Figure 14.2 An illustration of a sliding window with a fixed size of three. When three packets are outstanding, i.e., have not been acknowledged, transmission has to pause. As each acknowledgement is received, the window is slided by one packet, allowing a new packet to be transmitted.

so-called *self-clocking property* driven by the acknowledgement packets. A picture illustrating the sliding-window operation is shown in Figure 14.2.

- *Additive increase and multiplicative decrease.* We will not have the time to discuss the details of how the `cwnd` value is initialized as a new TCP connection is established, during the so-called **slow start** phase. We focus on the **congestion avoidance** phase instead. If there is no congestion, `cwnd` should be allowed to grow, in order to efficiently utilize link capacities. *Increasing* the `cwnd` value is different from *sliding* the window under the same given `cwnd` value: `cwnd` becomes larger in addition to getting slided forward. And in TCP, when `cwnd` grows, it grows *linearly*: `cwnd` is increased by 1/`cwnd` upon receiving each acknowledgement. That means that over one round trip time, `cwnd` grows by 1 if all ACKs are properly received. This operation is shown in the space–time diagram in Figure 14.3. But if there is congestion, `cwnd` should be reduced so as to alleviate congestion. And TCP says when `cwnd` is cut, it is cut *multiplicatively*: `cwnd` next time is, say, half of its current value. Increasing `cwnd` additively and decreasing it multiplicatively means that the control of packet injection into the network is conservative. It would have been much more aggressive if it were the other way around: multiplicative increase and additive decrease.
- *Infer congestion by packet loss or delay.* But how do you know whether there is congestion? If you are an iPhone running a TCP connection, you really have no idea what the network topology looks like, what path your packets are taking, which other end hosts share links with you, and which links along the path are congested. You have only a local and noisy view, and yet you have to make an educated guess: is your connection experiencing

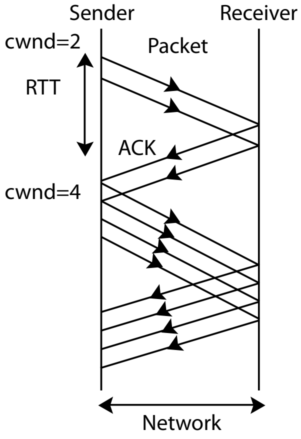


Figure 14.3 A space–time diagram of TCP packets being sent and acknowledged. The horizontal distance between the two vertical lines represents the spatial distance between the sender and the receiver. The vertical axis represents time. As two acknowledgements are received by the sender, the congestion window not only slides, but also increases by 1.

congestion somewhere in the network or not? The early versions of TCP congestion control made an important assumption: if there is a packet loss, there is congestion. This sounds reasonable enough, but sometimes packet loss is caused by a bad channel, like in wireless links, rather than congestion. In addition, often it is a little too late to react to congestion by the time packets are already getting dropped. The first problem has been tackled by many proposals of TCP for wireless. The second problem is largely solved by using packet delay as the congestion feedback signal. Instead of a binary definition of congestion or no congestion, a delay value implies the *degree* of congestion.

- *Estimate packet loss and delay by timers.* Say you agree that packet loss or delay implies congestion, how can you tell whether a packet has been lost and how do you calculate delay? TCP uses two common sense approximations. (1) If the sender waits for a long time and the acknowledgement does not come back, probably the packet has been lost. How long is a "long time"? Say this timeout timer is three times the normal **round trip time** (RTT) between the sender and the receiver. And what is the "normal" RTT? The sender timestamps each packet, and can tell the RTT of that packet once the acknowledgement is received at a later time. This is how the sender calculates a delay for each packet. Then it can calculate a moving-averaged RTT. The smallest RTT over a period of time is approximately the no-congestion, "normal" RTT. (2) Each packet sent has a sequence number, and if the sender hears from the receiver that several, say three, later packets (numbered 10, 11, and 12) have been received but this particular packet 9 is still not yet received, that probably means packet 9 has been lost. Packet 9 may have traversed a different path with a longer RTT (as discussed in IP routing in the last chapter), but if as many as three later packets have already arrived, chances are that packet 9 is not just late but lost.

As mentioned in the last chapter, TCP/IP is the “thin waist” of the Internet layered protocol stack. It glues the functional modules below it, like the physical and link layers, to those above it, like the application layer. (There are alternatives to TCP in this thin waist, such as the connectionless UDP that does not maintain an end-to-end feedback control that we will see in Chapter 17.) As part of that thin waist, the above five elements of congestion control design in TCP led to a great success. The wonderful fact that the Internet has not collapsed, despite the incredible and unstoppable surge of demand, is partially attributable to its congestion control capability.

Starting with TCP Tahoe in 1988 and its slightly modified variant **TCP Reno** in 1990, TCP congestion control had gone through over twenty years of improvement. For example, **TCP Vegas** in 1995 shifted from a loss-based congestion signal to a delay-based congestion signal. **FAST TCP** in 2002 stabilized congestion control to achieve high utilization of link capacity. **CUBIC** in 2005 combined loss- and delay-based congestion signals, and is now the default TCP in the Linux kernel. There have also been many other variants of TCP congestion control proposed over the past two decades.

14.1.2 Loss-based congestion inference

If you think about it, for end-to-end congestion control without any message passing from the network, an end host (like your iPad) really has very little to work with. Estimates of packet loss and calculations of packet delay are pretty much the only two pieces of information it can obtain through time stamping and numbering the packets.

For loss-based congestion control like TCP Reno, a major TCP variant especially for the Windows operating system, the main operations are as follows.

- If all the `cwnd` outstanding packets are received at the receiver properly (i.e., in time and not out of order more than twice), increase the `cwnd` by 1 each RTT.
- Otherwise, decrease it by cutting it in half, e.g., from `cwnd` to 0.5×`cwnd`.

There are also other subtle features like Fast Retransmit and Fast Recovery that we will not have time to get into.

Let us look at an example. For simplicity, let RTT = 1 unit, and assume it is a constant. Actually, RTT is about 50 ms across the USA and varies as the congestion condition changes. Initialize `cwnd` to be 5. Suppose all packets

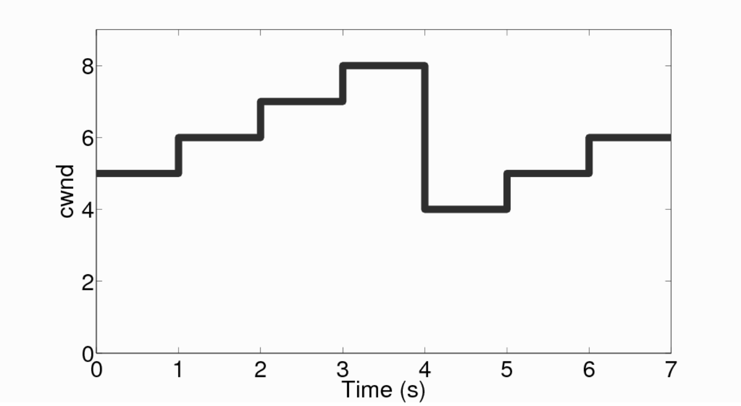


Figure 14.4 A zoomed-in view of `cwnd` evolution for TCP Reno, with RTT=1 unit of time.

are successfully received and acknowledged (ACK) during each RTT, except at $t = 4$, when a packet loss occurs.

At $t = 0$, `cwnd`=5, so the sender sends five packets and pauses.

At $t = 1$, the sender has received five ACKs, so it slides the congestion window by five packets and increases `cwnd` by 1. It sends six packets.

At $t = 2$, the sender has received six ACKs, so it sends seven packets.

At $t = 3$, the sender has received seven ACKs, so it sends eight packets.

At $t = 4$, the sender detects a lost packet. It halves `cwnd` to four, and sends four packets.

At $t = 5$, the sender has received four ACKs, so it sends five packets.

At $t = 6$, the sender has received five ACKs, so it sends six packets.

Figure 14.4 shows these values of `cwnd` over time. When there was no packet loss ($t = 0, 1, 2, 3$), `cwnd` grew linearly. When the packet loss occurred ($t = 4$), `cwnd` decreased sharply, then began growing linearly again ($t = 5, 6$).

Zooming out, Figure 14.5(a) shows a typical evolution of TCP Reno's `cwnd` over time. The y-axis is the congestion window size. If you divide that by the RTT and multiply it by the average packet size, you get the actual transmission rate in bps.

14.1.3 Delay-based congestion inference

Now we turn to delay-based congestion control like TCP Vegas. We first have to appreciate that the total RTT is mostly composed of **propagation delay**, the time it takes to just go through the links, and **queueing delay**, the time a packet spends waiting in the queue due to congestion. The heavier the congestion, the longer the wait. So the sender needs to estimate RTT_{min}, the minimum RTT that tells the sender what the delay value should be if there is (almost) no congestion.

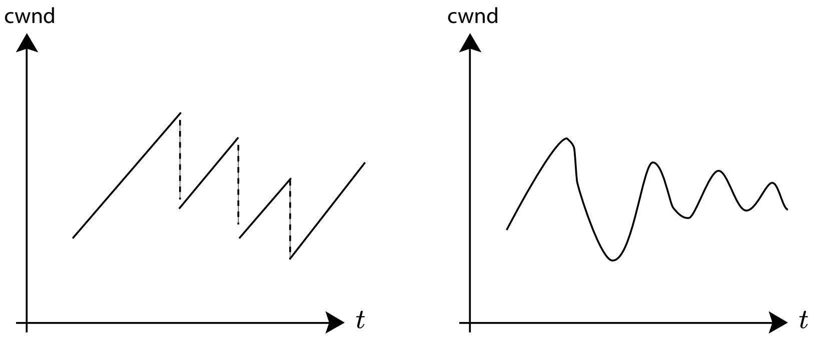


Figure 14.5 Typical evolutions of `cwnd` values in TCP Reno on the left and in TCP Vegas on the right. TCP Reno uses loss as the congestion signal whereas TCP Vegas uses delay as the congestion signal. The zig-zags between overshooting and under-utilizing capacity tend to be smaller in Vegas if the parameters are properly tuned.

Then, upon receiving each acknowledgement, the sender looks at the difference between `cwnd`$/RTT_{min}$ and `cwnd`$/RTT_{now}$. It is the difference between the transmission rate (in packets per second) without much congestion delay and that with the current congestion delay.

- If this difference is smaller than a prescribed threshold, say 3, that means there is little congestion, and `cwnd` is increased by 1.
- If the difference is larger than the threshold, that means there is some congestion, and `cwnd` is decreased by 1.
- If the difference is exactly equal to the threshold, `cwnd` stays the same.
- If all sources stop adjusting their `cwnd` values, an equilibrium is reached.

We can compare this congestion control with the power control in Chapter 1: at the equilibrium everyone stops changing their variable simultaneously. We would like to know what exactly the resource allocation is at such an equilibrium, and whether it can be reached through some simple, distributed, iterative algorithm.

Figure 14.5(b) shows a typical evolution of TCP Vegas' `cwnd` over time. You can see that the zig-zag between a rate that is too aggressive (leading to congestion) and one that is overly conservative (leading to under-utilization of link capacities) can be reduced, as compared with TCP Reno. Using delay as a *continuous* signal of congestion is better than using only loss as a *binary* signal, and we will see several arguments for this observation in the next section.

14.2 A Long Answer

Whether distributedly like TCP or through a centralized command system, any protocol trying to control congestion in a network must consider this fundamental

15 How can Skype and BitTorrent be free?

We just went through some of the key concepts behind the TCP/IP thin waist of the Internet protocol stack. We will now go through five more chapters on technology networks, focusing on two major trends: massive amounts of content distribution and the prevalent adoption of mobile wireless technologies.

Scaling up the distribution of content, including video content, can be carried out either through the help of peers or by using large data centers. These two approaches, P2P and cloud, are described in this chapter and the next, respectively. In particular, P2P illustrates a key principle behind the success of the Internet: under-specify protocols governing the operation of a network so that an overlay network can be readily built on top of it for future applications unforeseen by today's experts. It also illustrates the importance of backward compatibility, incremental deployability, and incentive alignment in the evolution of the Internet.

15.1 A Short Answer

Skype allows phone calls between IP-based devices (like laptops, tablets, and smartphones) or between IP devices and normal phones. It is free for IP-to-IP calls. How could that be? Part of the answer is that it uses a peer-to-peer (**P2P**) protocol riding on top of IP networks.

P2P started becoming popular around 1999. For example, Kazaa and Gnutella were widely used P2P file- and music-sharing systems back then. However, incentives were not properly designed in those first-generation P2P systems; there were a lot of *free riders* who did not contribute nearly as much as they consumed.

Skype started in 2001 from Kazaa, and was acquired by eBay for $2.6 billion in 2006 and then by Microsoft for $8 billion in 2011. As of 2010, there were 663 million Skype users worldwide. On any given day there are, on average, 700 million minutes of Skype calls.

BitTorrent started in 2001 as well, and is heavily used for file sharing, including movie sharing. Like Skype, it is free and uses P2P technologies. At one point, P2P was more than half of Internet traffic, and BitTorrent alone in the mid 2000s was about 30% of the Internet traffic. P2P sharing of multimedia content is still very popular today, with over 250 million users just in BitTorrent.

P2P showcases a major success of the evolution of the Internet: make the basic design simple and allow overlay constructions. The architecture of the Internet focuses on providing simple, ubiquitous, stable, and economical connectivities, leaving the rest of the innovations to overlays to be constructed in the future for unforeseeable applications. Different types of applications, unicast as well as multicast, have been built using P2P overlays, including file sharing, video streaming, and on-demand multimedia distribution.

Both Skype and BitTorrent are free (of course the Internet connection from your device might not be free).

- Skype is free in part because it leverages peer capability to locate each other and establish connections. P2P is used for signaling in Skype.
- BitTorrent is free in part because it leverages peer uplink capacities to send chunks of files to each other, without deploying many media servers. (And it is free in part because the content shared sometimes does not incur royalty fees). P2P is used for sharing content in BitTorrent.

Both Skype and BitTorrent are *scalable*. They illustrate a positive network effect whereby each additional node in the network contributes to many other nodes. We can therefore add many more nodes as the network scales up without creating a bottleneck. Of course this assumes the nodes can effectively contribute, and that requires some smart engineering design. As this chapter shows, this "**P2P law**" is a refinement of our intuition about the network effect codified in **Metcalfe's law** (named after the inventor of Ethernet that connects computers in a local area network): the benefit of joining a network grows as the *square* of the number of nodes. One of the underlying assumption is that all connections are equally important. But as we saw in Chapter 9 on triad closures vs. long-range links and in Chapter 10 on long-tail distribution, there is often a "diminishing marginal return" on similar types of links. Metcalfe's law also makes an assumption that each node is basically connected to all the other nodes, or at least the number of neighbors per node grows as a linear function of the network size. In contrast, the P2P law does *not* require that, and shows that the benefit of scalability can be achieved even when each node has only a small number of neighbors at any given time, as long as these are carefully chosen.

Skype's operational details are a commercial secret. BitTorrent is much more transparent, with papers written by the founder explaining its operation. So our treatment of Skype P2P connection management will be thinner than that of BitTorrent's P2P content sharing.

15.1.1 Skype basics

To understand how the technology behind Skype works, we need to understand two major topics: voice over IP (**VoIP**) and P2P. We postpone the discussion of VoIP to Chapter 17 together with multimedia networking in general. This chapter's focus is on P2P.

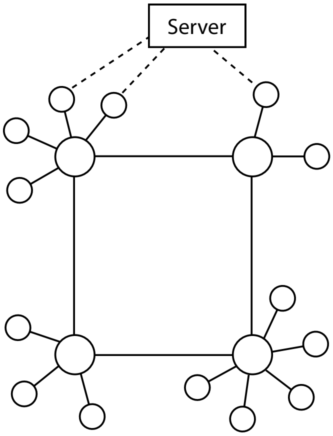


Figure 15.1 A typical topology of Skype. There is a mesh of super nodes (the bigger circles) and a shallow tree of ordinary nodes (smaller circles) rooted at each super node. Super nodes can be users' computers or skype machines. There is also an authentication server (the rectangle) that each node exchanges control messages with first.

Phone calls are intrinsically P2P: a peer calls another peer (as opposed to a server). What is interesting is that Skype uses P2P to discover peers and to traverse firewalls (software and hardware that blocks incoming data connections). As shown in Figure 15.1, Skype's central directory allows a caller to discover the IP address of the callee and then establish an Internet connection. These directories are replicated and distributed in **super nodes** (SNs).

The problem is that sometimes both the caller and the callee are behind firewalls, with a NAT box (see Chapter 13) in between. So the actual IP address is not known to the caller. Those outside of a firewall cannot initiate a call into the firewall.

What happens then is that super nodes have *public* IP addresses, serving as anchors to be reached by anyone and collectively acting as a network of publicly visible relays. The caller first initiates a connection with an SN, and the callee initiates a connection with another SN. Once a connection has been established, two-way communication can happen. The caller then calls her SN, which calls the callee's SN, which then calls the callee. Once a connection between the caller and the callee has been established through these two SNs, they mutually agree to use just a single SN that they both can connect to, thus shortening the communication path.

15.1.2 BitTorrent basics

BitTorrent uses P2P for resource sharing: sharing upload capacities of each peer and sharing the content stored in each peer, so that content sharing can scale

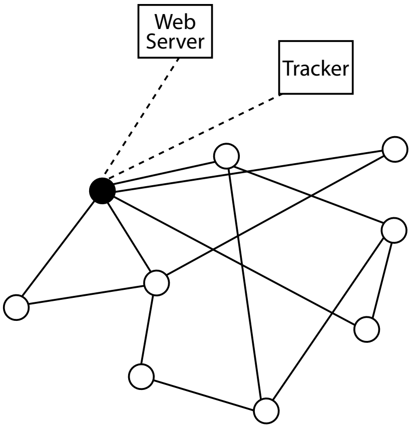


Figure 15.2 A typical topology of BitTorrent. There are actually three topologies: (1) a graph of physical connections among peers and routers, (2) a graph of overlay neighbor relationships among peers, and (3) a graph of peering relationships among peers. Graph (3) is an overlay on graph (2), which is in turn an overlay on (1). This figure shows graph (3). It changes regularly depending on the list of peers provided by the tracker to, say, peer A (the dark circle), as well as the subset of those peers chosen by peer A.

itself. It is designed primarily for **multicasting**: many users all demand the same file at about the same time.

In BitTorrent, each file is divided into small pieces called *chunks*, typically 256 kB, so that pieces of a file can be shared simultaneously. Each peer polls a centralized directory called the *tracker*, which tells a peer a set of 50 (or so) peers with chunks of the file it needs. Then the peer picks five peers with which to exchange file chunks. This set of five peering neighbors is refreshed at the beginning of every timeslot, depending in part on how much a neighbor is helping this peer and in part on randomization.

As shown in Figure 15.3, each individual chunk traverses a *tree* of peers, although the overall peering relationship is a general graph that evolves in time. A **tree** is an undirected graph with only one path from one node to any other node. There are no cycles in a tree. We usually draw a tree with the root node on top and the leaf nodes on the bottom.

We see that the control plane for signaling is somewhat centralized in both Skype and BitTorrent, but the data plane for the actual data transmission is distributed, indeed peer to peer. This is in sharp contrast to the traditional **client–server** architecture, where each of the receivers requests data from a centralized server and do not help each other.

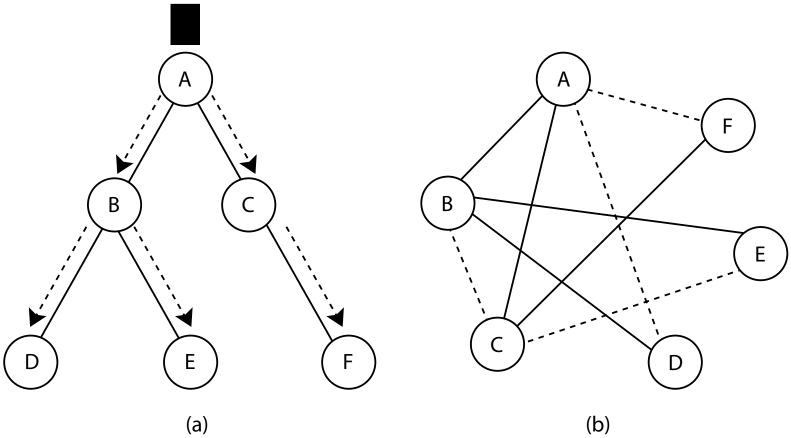


Figure 15.3 (a) Each chunk traverses a tree (with the chunk represented by the rectangle and the data transmission in dotted lines), even though (b) the peering relationships form a general graph (where the solid lines represent the current peering relationships and dotted lines represent possible peering relationships in the next timeslot).

15.2 A Long Answer

Before we go into some details of the smart ideas behind Skype and BitTorrent, we highlight two interesting observations.

- P2P is an **overlay network**, as illustrated in Figure 15.4. Given a graph with a node set V and a link set E, $G = (V, E)$, which we call the underlay, if we select a subset of the nodes in V and call that the new node set $\tilde{V}$, and we take some of the *paths* connecting nodes in $\tilde{V}$ as *links* and call that the new link set $\tilde{E}$, we have an overlay graph $\tilde{G} = (\tilde{V}, \tilde{E})$. The Internet itself can be considered as an overlay on top of the PSTN, wireless, and other networks; and online social networks are an overlay on top of the Internet too. The idea of overlay is as powerful as that of layering in giving rise to the success of the Internet. It is evolvable: as long as the Internet provides the basic service of addressing, connectivity, and application interfaces, people can build overlay networks on top of existing ones. For example, multicasting could have been carried out in the network layer through **IP multicast**. And there are indeed protocols for that. But other than within a Local Area Network (see the homework problem in Chapter 13) and IPTV for channelized content (see Chapter 17), IP multicast is rarely used. The management of IP multicast tends to be too complicated. Instead, P2P offers an alternative, overlay-based approach with less overhead.
- P2P is about *scalability*, and, in BitTorrent's case, scalability in *multicasting.* If you consume, you also need to contribute. This upside of the network effect is the opposite of the wireless network interference problem, where

16 What's inside the cloud of iCloud?

16.1 A Short Answer

In June 2011, Apple announced its iCloud service. One of the eye-catching features is its digital rights management of music content. The other part is its ability to let you essentially carry your entire computer hard drive with you anywhere and stream music to any device.

Cloud is more than just storage. For example, in the same month, Google introduced ChromeBook, a "cloud laptop" that is basically just a web browser with Internet connectivity, and all the processing, storage, and software are somewhere in Google servers that you access remotely.

These new services and electronics intensify the trends that started with web-based emails (e.g., Gmail), software (e.g., Microsoft Office 365), and documents (e.g., Google Docs and Dropbox), where consumers use the network as their computers, the ultimate version of online computing.

In the enterprise market, many application providers and corporations have also shifted to cloud services, running their applications and software in rented and shared resources in **data centers**, rather than building their own server farms. Data centers are facilities hosting many servers and connecting them via many switches. Large data centers today is typically over 300,000 square feet, house half a million servers, and cost hundreds of millions of dollars to build.

There are three major cloud providers as of 2012: Amazon's EC2, Microsoft's Azure, and Google's AppEngine. A pioneering player in cloud services is actually Amazon, even though to most consumers Amazon stands for an online retail store. In Amazon's S3 cloud service today, you can pay $0.115 per hour for a small standard instance, featuring 1 virtual core, 1 EC2 unit, and 160 GB of storage. Any amount of data coming into the EC2 is free, and for outgoing data, each month the first GB is free, and the next 10 TB is $0.120 per GB.

For many years, it has been a goal in the computing and networking industries that one day users could readily rent resources inside the network (the "cloud" on a typical network illustration), in a way that makes economic sense for all the parties involved. That day is today. Thanks to both technological advances and new business cases, cloud services are taking off and evolving fast.

Many features of cloud services are not new, some are in fact decades-old. Several related terms have been used in the media somewhat confusingly too:

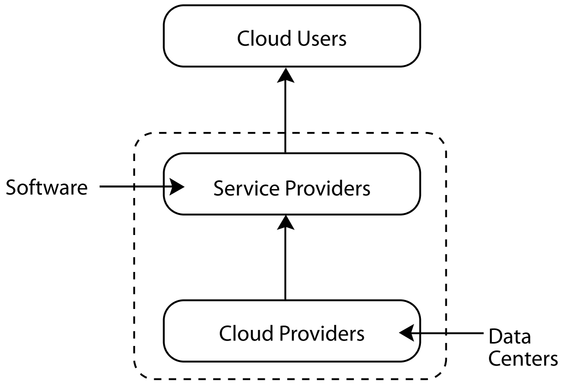


Figure 16.1 Three segments of the cloud service industry. Cloud providers operate data centers that house the hardware, including inter-connected processors, storage, and network capacity. Service providers run cloud services through their software. Cloud users include both consumer and enterprise customers for cloud services provided collectively by the service providers and cloud providers.

cloud computing, utility computing, clustered computing, software as a service, etc. To clarify the terminology, we refer to the graph in Figure 16.1. There are three key components of the "food chain."

Cloud providers build and manage the hardware platform, consisting of computing resources (servers), networking resources (switches), and storage resources (memory devices) organized inside data centers. There is a network within each data center where the nodes are servers and switches, and each data center in turn becomes a node in the Internet.

Service providers offer software and applications that run in data centers and interface with users. For example, an iPad application developer may use the computing and storage resources in Amazon's EC2 cloud to deliver its services. Sometimes, the service provider is the same as the cloud provider. For example, the iCloud music storage and streaming service from Apple runs in Apple's own data centers.

Cloud users are consumers and enterprises that use services running in data centers. Users can get the content they want (e.g., documents, books, music, video) or the software they need (e.g., Office software, an iPhone application, or in the cloud laptop's case, pretty much any software you need) from the cloud. And get them on demand, anytime, anywhere, and on any device with an Internet connection.

16.1.1 What features define cloud services?

To make the overall cloud service food chain work, we need all of the following ingredients:

1. *Large-scale computing and storage systems*, often leveraging virtualization techniques in sharing a given hardware resource among many processes as if they each had a slice of a dedicated and isolated resource.
2. *Networking* within a data center, across the data centers, and to the end-users (often with a wireless hop like WiFi or 4G). This networking dimension naturally will be the focus of our discussion in this book, especially networking within a data center.
3. *Software* that provides a graphical user interface, digital rights management, security and privacy, billing and charging, etc.

If I open up my home desktop's CPU and hard drive to renters, does that constitute a cloud service? Probably not. So, what are the defining characteristics of a cloud service? The keyword is *on demand*, along two dimensions: time and scale.

- On demand in timing: a cloud service allows its users to change their requests for resources at short notice, and possibly only for a short period of time.
- On demand in scale: a cloud service allows its users to start at a very small minimum level of resource request (e.g., 1.7 GB of RAM and 160 GB of memory on Amazon's EC2 today), and yet can go to really large scale (e.g., Target, the second-largest retail store chain in the USA, runs its web and inventory control in a rented cloud).

16.1.2 Why do some people hesitate to use cloud services?

Cloud services face many challenges, even though they are increasingly outweighed by the benefits. Let us briefly bring them up before moving on.

Similarly to the pros–cons comparison between packet switching and circuit switching, once you are in a shared facility, the performance guarantee is compromised and so are security and privacy. If you ride a bus instead of a taxi, you pay less but you might not have a seat and you will be seen by the fellow bus riders. That is the price you pay to enjoy the benefits of cloud services. There are many technologies that mitigate cloud's downsides for various market segments, but riding a bus will never be exactly the same as taking a taxi.

As illustrated by the Amazon cloud outage in April 2011 and more recently the storm-induced electricity outage to some of the Netflix service in June 2012, availability of service in the first place is the top performance concerns. The main root causes for unavailability include network misconfigurations, firmware bugs, and faulty components. The best way to enhance availability is *redundancy*: spread your traffic across multiple cloud providers (assuming it is easy enough to split and merge the traffic), and across different reliability zones in each of the providers.

16.1.3 Why do some people like cloud services?

Why does it make sense to provide and to use cloud services? The answers are similar to those relating to many other rental businesses, such as libraries and rental car companies. We summarize the arguments below and will go through an example in a homework problem.

To the cloud users, the key benefit is *resource pooling*. The cost of building and provisioning resources is now shared by many other users. This is called the "CapEx to OpEx conversion:" instead of spending money in capital expenditure to build out dedicated facilities, a user pays rent as part of its operational expenditure to share facilities. This is analogous to going from circuit switching to packet switching in the design of the Internet. The risk of miscalculating resource need shifts to cloud providers, a significant advantage if the resource demand varies a lot or is just hard to predict.

But why would cloud *providers* be interested? A main reason is the *economy-of-scale* advantages, both on the supply side and on the demand side of the business. On the supply side, a cloud provider can procure the servers, switches, labor, land, and electricity at significantly discounted prices because of its large-scale and bargaining power. Even when compared with a medium-sized data center with thousands of servers, a large scale data center with a hundred thousand servers can often achieve a factor of $5-7$ advantage in cost per GB of data stored or processed.

On the demand side, scale helps again, through *statistical multiplexing*. Fluctuations of demand for each user are absorbed into a large pool of users, as

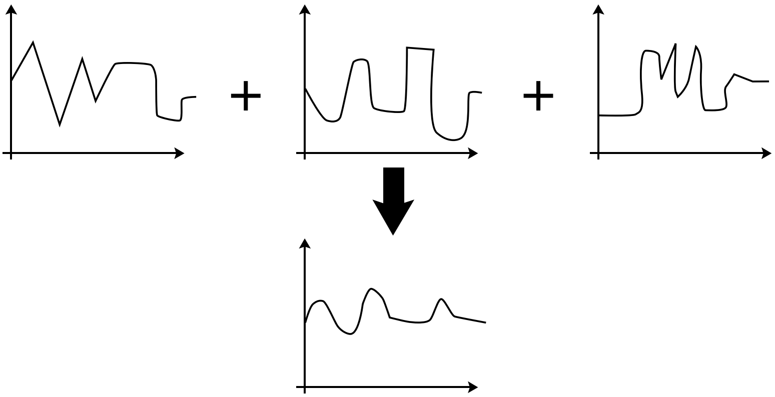


Figure 16.2 Statistical multiplexing smoothes out the burstiness of individual users. Suppose there are three users with their transmission rates over time as charted above. Their aggregate transmission rate, shown in the lower graph, is much smoother. Cloud providers leverage burstiness reduction as the scale goes up to reduce capacity provisioning, as long as the users' demands do not peak together. The scale of the lower graph's y-axis is about three times that of the upper graphs'.

illustrated in Figure 16.2. This is the same principle as that behind ISPs oversubscribing at each aggregation point of their access networks: aggregating many bursty users reduces the burstiness. Of course, the overall pool may still exhibit time-of-day peak–valley patterns. The *average* utilization of servers in a data center is often below 20% today. These peak–valley patterns can be further smoothed out by time-dependent pricing as discussed in Chapter 12.

Cloud is all about *scale*. Today's large-scale data centers are indeed huge, so big that electricity and cooling costs sometimes amount to more than half of the total cost. If an iPhone is one of the smallest computers we use, each data center is one of the largest. We have made an important assumption, that it is actually *feasible* to scale up a data center. Otherwise, we would have to truncate all the benefits associated with scaling up. But, as we saw in Chapter 10, scale can also be a *disadvantage* when each (reasonably priced) network element can have only a small number of high-performance ports. Unless you have the right network topology, building a 100,000-server data center can be much more expensive, in unit price of capacity (or, "bandwidth" in this field's common terminology), than building a 10,000-server data center. This echoes Chapter 10's theme: (high-throughput) connectivity per node does *not* scale up beyond a certain point in either technology or human networks. Yet we want (high-throughput) connectivity in order for the whole network to keep scaling up. That is the subject of the next section: how to achieve the advantages of scale for a network without suffering the limitation of scale of each node.

16.2 A Long Answer

16.2.1 Building a big network from small switches

We need a network within a data center. Many applications hosted in a data center require transfer of data and control packets across the servers at different locations in that big building. A natural, but inferior solution, is to build a tree like Figure 16.3(a), where the leaf nodes are the servers, and the other nodes are the routers. The low-level links are often 1 Gbps Ethernet, and upper level ones 10 Gbps Ethernet links. The top-of-the-tree switches are big ones, each with many 10 Gbps links. It is expensive to build these big switches. As the number of leaf nodes increases to 100,000 and more, it becomes technologically impossible to build the root switch. A high-end switch today can support only 1280 servers.

So we need to start *oversubscribing* as we climb up the tree. Sometimes the **oversubscription ratio** runs as high as 1:200 in a large data center. What if all the leaf-node servers want to fully utilize their port bandwidths to communicate with other servers at the *same* time? Then you have a 200-factor congestion. The whole point of resource pooling is defeated as we "fragment" the resources: idle servers cannot be utilized because the capacity between them cannot be used in

17 IPTV and Netflix: How can the Internet support video?

We saw in Chapter 13 that the Internet provides a "best effort," i.e., "no effort" service. So, how can it support video distribution that often imposes stringent demands on throughput and delay?

17.1 A Short Answer

17.1.1 Viewing models

Watching video is a significant part of many people's daily life, and it is increasingly dependent on the Internet and wireless networks. Movies, TV shows, and home videos flow from the cloud through the IP network to mobile devices. This trend is changing both the networking and the entertainment industries. As of 2011, there were more than 100 million IPTV users in the USA, and Youtube and Netflix together take up about half of the Internet capacity usage. As the trend of decoupling among contents, content delivery channels, and content-consuming devices intensifies, IP has become the basis of almost all the content distribution systems.

This trend is bringing about a revolution in our viewing habits.

- *Content type*: Both user-generated and licensed content have become prevalent. Clearly, more user-generated content implies an increasing need for upload capacity, which is traditionally designed to be much smaller than download capacity.
- *When*: For many types of video content, we can watch them anytime we want, with the help of devices like a Digital Video Recorder (DVR) on IPTV or services like HBO Go.
- *Where*: We can watch video content almost anywhere, at least anywhere with a sufficiently fast Internet connection.
- *How*: Instead of just on the TV and desktop computers, we can watch video on our phones, tablets, and any device with a networking interface and a reasonable screen.
- *How much*: We are watching more video, thanks to applications like Netflix, Hulu, Deja, and embedded videos on many websites. For example, in February 2012, Hulu had roughly 31 million unique viewers, watching 951 million

videos. Comcast NBS Universal had 39 million unique viewers, watching 205 million videos (more than doubling the number from summer 2011). Some of these are free, some are free but with intrusive advertisements, some require a monthly subscription, some are pay-per-view, and some are part of a bundled service (e.g., the triple play of IPTV, Internet access, and VoIP). If the Internet connection charge is usage-based, there is also the "byte-transportation" cost per GB, as discussed in Chapter 11.

We can categorize viewing models along four dimensions. Each combination presents different implications to the network design in support of the specific viewing model.

- *Real time vs. precoded*: Some videos are watched as they are generated in real time, e.g., sports, news, weather videos. However, the vast majority are precoded: the content is already encoded and stored somewhere. In some cases, each video is stored with hundreds of different versions, each with a different playback format or bit rate. Real-time videos are more sensitive to delay, while precoded videos have more room to be properly prepared. Some other video-based services are not only real-time, but also two-way interactive, e.g., video calls, video conferencing, and online gaming. Clearly, interactive video has even more stringent requirements on delay and jitter (i.e., the variance of delay over time).
- *Streaming or download*: Some videos, like those on Netflix and YouTube, are streamed to you, meaning that your device does not keep a local copy of the video file (although Netflix movies sometimes can be stored in a local cache, and YouTube has started a movie-rental service). In other cases, e.g., iTunes, the entire video is downloaded first before played back at some later point. Of course, the content itself may be automatically erased from local storage if digital rights are properly managed, like in movie rentals. In-between these two modes there is the possibility of *partial* download and playback. As shown in the Advanced Material, this reduces the chance of jitter, and is followed in practice almost all the time except for interactive or extremely real-time content.
- *Channelized or on-demand*: Some contents are organized into channels, and you have to follow the schedule of each channel accordingly. This is the typical TV experience we have had for decades. Even with DVR, you still cannot jump the schedule in real time. In contrast, Video on Demand (VoD) allows you to get the content when you want it. Both YouTube and Netflix are VoD. There are also VoD services on TV, usually charging a premium. Sometimes the content owner changes the model, e.g., in 2011 HBO in the USA changed to a VoD model with its HBO Go services on computers and mobile devices. In-between the two extremes, there is *NVoD*, Near Video on Demand, which staggers the same channel every few minutes, so that, within a latency tolerance of that few minutes, you get the experience of VoD.

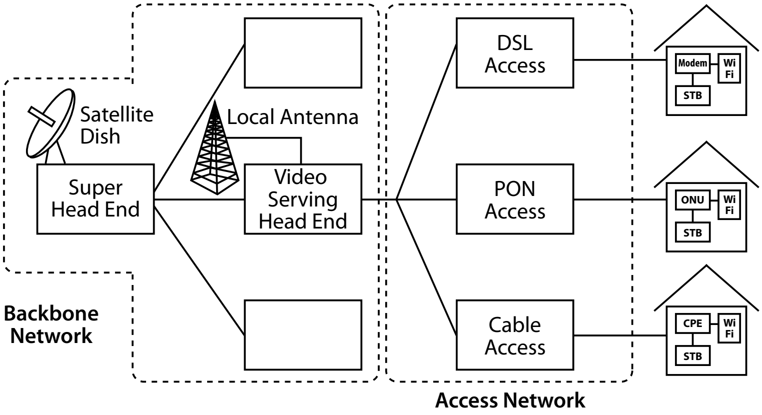


Figure 17.1 A typical architecture of IPTV. The content is collected at the super head end and distributed to different local video-serving head ends across the country, which also collect local content. Then it is further distributed to access networks running on copper, fiber, or cable, before reaching the homes. This is often carried out in private networks owned and managed by a single ISP.

- *Unicast or multicast*: Unicast means transmission from one source to one destination. Multicast means from one source to many destinations, possibly tens of millions for events like the Olympic Games, that belong to a multicast group. An extreme form of multicast is *broadcast*: everyone is in the multicast group. If you do not want certain content, you do not have to watch it, but it is sent to you anyway. TV is traditionally multicast, sometimes through physical media that are intrinsically multicast too, such as satellite. The Internet is traditionally unicast. Now the two ends are getting closer. We see unicast capabilities in IP-based video distribution, but also multicast in IP networks (carried out either in the network layer through IP multicast routing or in the application layer through P2P).

It seems that there are $2^4 = 16$ combinations using the above taxonomy of video viewing modes. Obviously some combinations do not make sense, for example, real-time video must be streaming-based and cannot be download-based. But precoded video can be either streaming- or download-based. Or, true VoD cannot be multicast since each individual asks for the content at different times, but channelized or NVoD content can be either unicast or multicast.

17.1.2 IP video: IPTV and VoI

The term “IP video” actually encompasses two styles: (1) IPTV and (2) VoI. IPTV turns TV channel delivery into IP-based, whereas VoI views the Internet as a pipe that can simply support any type of content delivery. Increasingly VoI

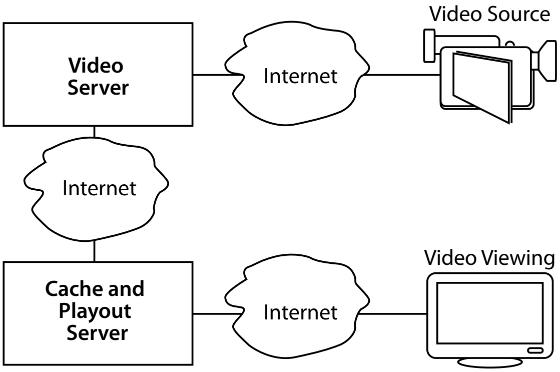


Figure 17.2 A typical architecture of video over the Internet. Video sources, ranging from iPhones to professional video cameras, upload content to video servers, which then distribute them through local caches to the viewers around the world who download the video to their devices. This is all carried out in the public Internet.

is becoming the more popular way for people to consume video content than IPTV. If all content is available on demand through VoI, what advantages does IPTV offer to the consumer experience?

(1) **IPTV** is often included as part of the triple- or quadruple-play service bundle provided by an ISP. It is delivered over a *private and managed network*, with a set-top box on the customer's premises. This private network uses IP as a control protocol, but many parts of it are deployed and operated by a single ISP, e.g., a telephone or cable company offering the Internet access. This makes it easier to control the quality of service. The content is often channelized, multicast, and streaming-based but with recording capability using DVR.

Before TV turned to the Internet access networks, it was delivered primarily through one of the following three modes: broadcast over the air, via satellites, or through cables. So why is the IPTV revolution happening now? There are a few key reasons.

- *Convergence*: almost everything else is converging on IP, including phone calls. Putting video on IP makes it a unified platform to manage.
- *Cost*: Having a uniform platform reduces the costs of maintaining separate networks.
- *Flexibility*: IP has demonstrated that one of its greatest strengths is the ability to support diverse applications arising in the future.
- *Compression* has got better and access network *capacity* has increased sufficiently that it has become possible to send HD TV channels.

(2) **Video over the Internet** (**VoI**) is delivered entirely over *public networks*, often via unicast, and to a variety of consumer devices. Given the current evolution of business models, VoI is increasingly taking over the IPTV business

as consumers access video content over the IP pipes without subscribing to TV services. There are three main types of VoI.

- The content owner sends videos through server–client architectures without a fee, e.g., YouTube, ABC, and the BBC.
- The content owner sends videos through server–client architectures with a fee, e.g., Netflix, Amazon Prime, Hulu Plus, and HBO Go.
- Free P2P sharing of movies, e.g., Bit Torrent and PPLive.

We touched upon the revenue models for IPTV and VoI. As to the cost models, they often consist of the following items.

- *Content*: The purchase of content-distribution rights. Popular and recent movies and TV series are naturally more expensive.
- *Servers*: The installation and maintenance of storage and computing devices.
- *Network capacity*: The deployment or rental of networking capacity to move content around and eventually deliver it to consumers.
- *Customer premises equipment*, such as set-top boxes and games consoles.
- *Software*: The software systems that manage all of the above and interface with consumers.

Whether it is IPTV or VoI, the quality measures depend on the bit rate, delay, and jitter. What kind of bit rates do we need for videos? It depends on a few factors, e.g., the amount of motion in the video, the efficiency of the compression methods, the screen resolution, and the ratio of viewing distance and screen size. But, generally speaking, the minimum requirement today is about 300 kbps. Below that, the visual quality is just too poor even on small screens. For standard-definition movies we need at least 1 Mbps, and a typical movie takes $1 - 2$ GB. For high-definition movies we need at least $6 - 8$ Mbps, and a typical movie takes $5 - 8$ GB. Truly HD video needs $20 - 25$ Mbps to be delivered. And the latest standard of UltraHD needs over 100 Mbps. As we will see in the next section, to make IP video work, we need technologies from both multimedia signal processing and communication networking.

17.2 A Long Answer

As shown in Figure 17.3, the overall protocol stack for IP video includes the following: MPEG over HTTP/SIP/IGMP/RTSP, over RTP/UDP/TCP, over IP, over ATM or Ethernet or WiFi, over wireless/fiber/DSL/cable. In this section, we go into some detail regarding the top three layers: compression, application, and transport, trying to highlight interesting networking principles beyond an "alphabet soup" of acronyms.

18 Why is WiFi faster at home than at a hotspot?

A crude answer is that interference management in WiFi does not scale well beyond several devices sharing one access point. When the crowd is big, the “tragedy of the commons” effect, due to mutual interference in the unlicensed band, is not efficiently mitigated by WiFi. To see why, we have to go into the details of WiFi’s medium access control in the link layer of the layered protocol stack.

18.1 A Short Answer

18.1.1 How WiFi is different from cellular

Since their first major deployment in the late 1990s, WiFi hotspots have become an essential feature of our wireless lifestyle. There were already more than a billion WiFi devices around the world by 2010, and hundreds of millions are added each year. We use WiFi at home, in the office, and around public hotspots like those at airports, in coffee shops, or even around street corners.

We all know WiFi is often faster than 3G cellular, but you cannot move around too fast on WiFi service or be more than 100 m away from an Access Point (**AP**). We have seen many letters attached to 802.11, like 802.11a, b, g, and n, shown on the WiFi AP boxes you can buy from electronic stores, but maybe do not appreciate why we are cooking an alphabet soup. We have all used hotspot services at airports, restaurants, hotels, and perhaps our neighbor’s WiFi (if it does not require a password), and yet have all been frustrated by the little lock symbol next to many WiFi network names that our smartphones can see but not use.

When Steve Jobs presented iPhone 4 in a large auditorium, that demo iPhone could not get on the WiFi. Jobs had to ask all the attendants to get off the WiFi. Afterwards his iPhone managed to access the WiFi. Is there some kind of limit as to how many users a given WiFi hotspot can support?

In June 2012, five leading cable providers in the USA announced that they would join their fifty thousand hotspots into a single WiFi service. One year before then, the South Korean government announced a plan to cover the entire city of Seoul, including every corner of every street, with WiFi service by 2015. If many WiFi users aggregate around one popular street corner, how many hotspots need to be created to take care of that demand? And, more importantly, how can this traffic be backhauled from the WiFi air-interface to the rest of the Internet?

At home, a residential gateway of the Internet access is often connected to a WiFi AP, which provides the in-home wireless connectivities to desktops, laptops, games consoles, phones, tablets, and even TV's set-top boxes using the latest high speed WiFi variant. As each home adds more WiFi devices, will the quality of connection be degraded, especially in high-rise multi-tenant buildings?

The answers to these questions are continuously being updated by academia and industry, and we already know quite a bit about WiFi architecture and performance. Officially, WiFi should be called the **IEEE 802.11** standard. It is part of the 802 family of standards on Local Area Networks prescribed by the IEEE. The .11 part of the family focuses on wireless LAN using the **unlicensed spectrum**.

You must have a license from the government to transmit in the frequency bands used by all generations of cellular networks. For 3G and 4G, governments around the world sold these spectral resources for tens of billions of dollars, sometimes through auctions as we saw in a homework problem in Chapter 2. This avoids having too many transmitters crowding and jamming into those frequency bands.

In contrast, governments around the world also leave some bands unlicensed and free, as long as your transmit power is not too high. For example, the Industry, Science, and Medical (ISM) frequency ranges in the S-band around 2.4–2.5 GHz and in the C band around 5.8 GHz. They were originally allocated for use in the three fields, as suggested by the name ISM, but the most widely used appliance in the ISM S-band, other than WiFi, is actually the microwave oven. That band works well to excite water molecules. There are also other wireless communication devices running on bluetooth, zigbee, etc. sharing the same ISM band. Handling interference on an unlicensed spectrum is a major challenge.

In the mid 1990s, as the 2G cellular industry took off, people started wondering whether they could create an alternative in a wireless network: use the small amount of power allowed in ISM and short-range communication (around 100 m outdoors, a transmission range one order of magnitude smaller than that for cellular) for mostly stationary devices. Because this is not a single-provider network, an industry forum was needed to test the inter-operability of all the devices. It was established in 1999 as the Wi-Fi Alliance, where Wi-Fi stands for "Wireless Fidelity" and is a catchier name than "IEEE 802.11b."

Many versions of WiFi standards were created by the IEEE 802 organization, starting with 802.11b that uses the 2.4 GHz band and can transmit up to 11 Mbps. This was followed by two other main versions: 802.11g that uses a more advanced physical layer coding and modulation to get to 54 Mbps in the 2.4 GHz band, and 802.11a that can also achieve up to 54 Mbps in the 5.8 GHz band. Some of these standards divide the frequency band into smaller blocks in Orthogonal Frequency Division Multiplexing (**OFDM**). In contrast to anti-resource-pooling in Paris Metro Pricing in Chapter 12, this resource fragmentation is motivated

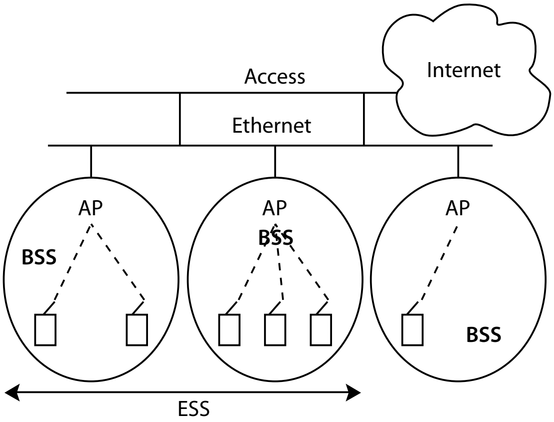


Figure 18.1 A typical topology of WiFi deployment. The air-interface provides bidirectional links between the APs and end user devices. Each BSS has an AP. A collection of BSSs that can readily support handoff is called an ESS. The air-interface is connected to a wireline backhaul, often an Ethernet, which is in turn connected to the rest of the access network and further to the rest of the Internet.

by better spectral efficiency because signals on smaller chunks of frequency bands can be more effectively processed. More recently, 802.11n uses multiple antennas on radios to push the transmission rate to over 100 Mbps. Augmenting the channel width to 40 MHz also helped increase the data rate. We will discuss OFDM and multiple antenna systems in the Advanced Material.

There have also been many supplements to improve specific areas of WiFi operation. For example, 802.11e improved the quality of service in its medium access control, a topic we will focus on in this chapter; 802.11h improved encryption and security, a major issue in the early days of WiFi; and 802.11r improved the roaming capability in WiFi to support, to some degree, the mobility of people holding WiFi devices.

Even though the nature of spectral operation is different, WiFi does share a similar topology (Figure 18.1) with cellular networks, except this time it is not called a cell (since there is often no detailed radio frequency planning before deployment), but a Basic Service Set (**BSS**). In each BSS there is an AP rather than a Base Station. A collection of neighboring BSSs may also form an Extended Service Set (ESS).

When your laptop or phone searches for WiFi connectivity, it sends probe messages to discover which APs are in its transmission range, and shows you the names of the BSSs. You might want to connect to a BSS, but, if it is password-protected, your device can associate with the AP only if you have the password to authenticate your status, e.g., as a resident in the building if the AP is run by the building owner, as an employee of the company if the AP is run by the corporation, or as a paying customer if the AP is part of the WiFi subscription service offered by a wireless provider. Increasingly you see more WiFi deployment, but *free* WiFi's availability may be on the decline.

These APs are tethered to a backhauling system, often a wireline **Ethernet** (another, and much older, IEEE 802 family member) that connects them to the rest of the Internet. This is conceptually similar to the core network behind the base stations in cellular networks, although the details of mobility support, billing, and inter-BSS coordination are often much simpler in WiFi.

If the channel conditions are good, e.g., you are sitting right next to the WiFi-enabled residential gateway at your home and no one else's signal interferes with yours, the data rate can be very impressive, especially if you are using 802.11n. It is faster than 3G, and probably even faster than the DSL or fiber access link that connects the residential gateway to the rest of the Internet. But if you sit outside the limited range of the AP or you start moving across the boundaries of an ESS, you can easily get disconnected. And if you share the air with ten or so other WiFi devices, the speed can drop substantially as you may have experienced in a crowded public WiFi hotspot.

There is actually also a peer-to-peer mode in 802.11 standards, the *infrastructureless*, ad hoc mode. WiFi devices can directly communicate with each other without passing through any fixed infrastructure like APs. You see this option when you configure the WiFi capability on your computers. But very few people use this mode today, and we will talk only about the infrastructure mode with APs.

18.1.2 Interference management in WiFi

Summarizing what we have talked about so far: WiFi is an evolving family of standards that enables short-range wireless communication over the ISM unlicensed bands for largely stationary devices. In contrast to cellular networks, WiFi networks are often deployed with very limited planning and managed only lightly, if at all.

WiFi is quite a different type of wireless networking from cellular, and its performance optimization requires some different approaches. Whether a WiFi hotspot works well or not really depends on how effectively such optimizations are carried out. We focus on the air-interface part between the AP and the devices (the terminology **station** covers both), even though the backhaul part could also become a bottleneck (e.g., when the DHCP server has a bug and cannot keep track of the IP addresses assigned to the devices, or simply because the backhauling capacity is limited.)

The first set of performance tuning involves the correct selection of the AP, the channel, and the physical layer transmission rate.

- *AP association*: A WiFi device has to regularly scan the air and then associate with the right AP, e.g., the one that offers the best SIR (and, of course, authenticates the device).

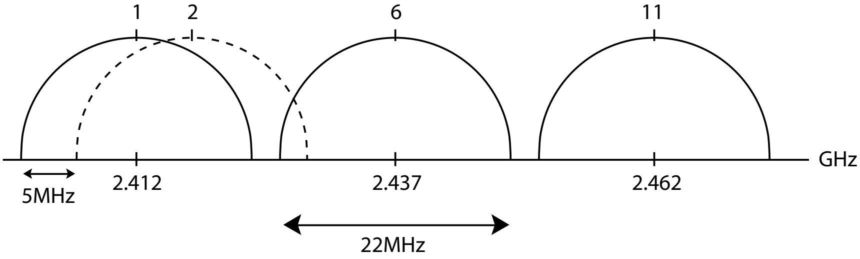


Figure 18.2 The 802.11b spectrum and channels. There are 11 channels in the USA. Each channel is 22 MHz wide, and 5 MHz apart from the neighboring channels. Therefore, only three channels, Channel 1, Channel 6, and Channel 11 are non-overlapping.

- *Channel selection*: The overall ISM frequency band is divided into channels. In 802.11b in the USA, for example, each channel is 22 MHz wide and 5 MHz apart from the neighboring channels. As shown in Figure 18.2, only those channels that are five channels apart are truly non-overlapping. So, if you want to have three devices on non-overlapping channels, the only configuration is for each of them to choose a different channel from among Channels 1, 6, and 11. Many WiFi deployments just use the default channel in each AP. If they are all on Channel 6, unnecessary interference is created right there.
- *Rate selection*: We mentioned that each of 802.11abgn can transmit *up to* a certain data rate. That is assuming a really good channel, with no interference and no mobility. In many WiFi hotspots, the channel condition fluctuates and interferers come and go. The maximum rate is rarely achieved, and the AP will tell the devices to backoff to one of the lower rates specified, e.g., all the way down to 1 Mbps in 802.11b, so that the decoding is accurate enough under the lower speed. A device knows it is time to reduce its rate to the next lower level if its receiver's SIR is too low for the current rate, or if there have been too many lost frames.

Suppose your WiFi device gets the above three parameters right. We need to take care of interference now. When two transmitters are within interference range of each other, and they both transmit a frame at similar time (t_1 and t_2, respectively), these two frames collide. There are three possible outcomes of a collision.

- Both frames are lost: neither receiver can correctly decode the intended frame.
- The stronger frame is properly received, but the weaker frame is lost: here, "strength" refers to SIR. This is called **capture**.
- Both frames are properly received. This is called **double capture**.

Now, which outcome will prevail? That depends, quite sensitively, on the following factors.

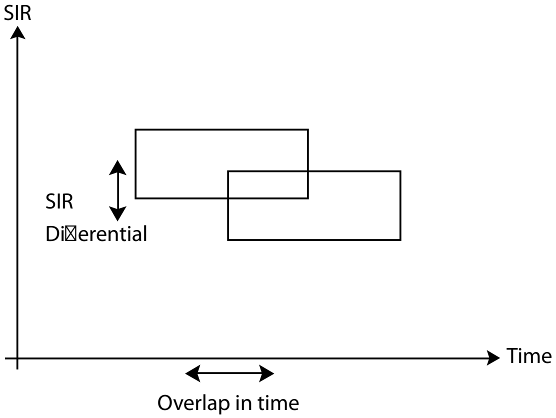


Figure 18.3 An energy-time diagram of two colliding frames. If collision is defined as two frames overlapping in their transmission time, the outcome of a collision depends quite sensitively on several factors: how long the overlap is, how big the differential in the received SIRs is, and how large an SIR is needed for proper decoding at each receiver.

- How long the frames overlap (based on their timing difference $t_1 - t_2$, frame sizes, and transmission rates).
- How big the differential in SIR between the two frames is (which depends on channel conditions and transmit powers). This is illustrated in Figure 18.3.
- How large an SIR is required for proper decoding at the receiver (which depends on transmission rates, coding and modulations, and receiver electronics).

It is an unpleasant fact that wireless transmissions may interfere with each other, since wireless transmission is energy propagating in the air. It is a particularly challenging set of physics to model, because collision is not just one type of event. In the rest of the chapter, we will simply assume that when a collision happens, both frames are lost.

Compared with power control in Chapter 1 for cellular networks, WiFi also uses a fundamentally different approach to manage interference, due to its much smaller cell size, the typical indoor propagation environment, a much smaller maximum transmit power allowed, and more uncontrolled interference in an unlicensed band. Instead of adjusting transmit powers to configure the right SIRs, WiFi tries to avoid collision altogether, through the mechanisms of **medium access control**.

Think of a cocktail party in Chapter 1 again, where guests' voices overlap in the air. With enough interference you cannot understand what your friend is trying to say. Cellular network power control is like asking each guest to adjust her volume without running into an arms race. WiFi medium access control is like arranging for the guests to talk at different times.

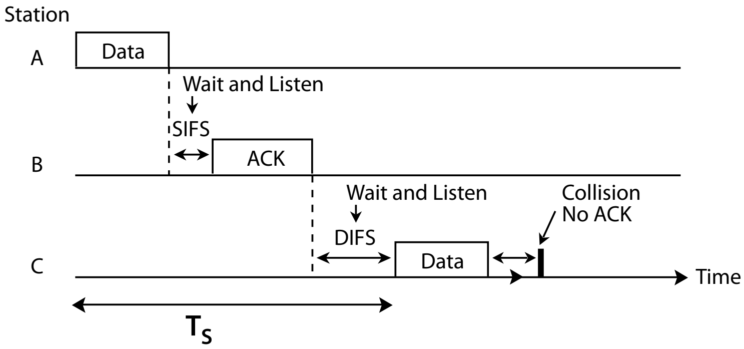


Figure 18.4 A timing diagram of basic WiFi transmissions. A station can be either a user device or an AP. First the transmitter of a session, station A, sends a data frame to its intended receiver, station B. Then, after a very short period of time with a predetermined length called the SIFS, B sends an acknowledgment frame back to A. After waiting for another slightly longer period of time called the DIFS, other nodes, like station C, can start sending new data frames. In the above example, node C's packet collides with some other packet transmitted by, say, station D.

You can either have a centralized coordinator to assign different timeslots for each guest to talk (scheduling), or you can ask each of them to obey a certain procedure for deciding locally when to talk and how long to talk (random access). We call these alternatives the Point Coordination Function (**PCF**) and the Distributed Coordination Function (**DCF**), respectively. The PCF, like a token ring in the wireline Ethernet protocol, represents centralized control (and dedicated resource allocation). It is complicated to operate and rarely used in practice. The DCF, in contrast, enables shared resource allocation. As you might suspect for any distributed algorithm, it can be less efficient, but easier to run. In practice, the DCF is used most of the time. We will discuss WiFi's DCF, which is a particular implementation of the Carrier Sensing Multiple Access (**CSMA**) random access protocol.

The basic operation of CSMA is quite simple and very intuitive. Suppose you are a transmitter. Before you send any frame, you regularly listen to the air (the part of the spectrum where your communication channel lies). This is called **carrier sensing**. As Figure 18.4 illustrates, every transmitter must observe a wait-and-listen period before it can attempt transmission. If the channel is sensed as busy (someone is using the medium to transmit her frames), you just stay silent. But if it is idle (no one is using it), you can go ahead and send a sequence of frames. You might want to send a lot of frames in a row, so you can send a control message declaring how long you intend to use the channel. Of course, the channel-holding time has an upper bound, just like in treadmill-sharing in a gym.

But if your frame collides with some other frames when you try to send it, your receiver will not get it (since we assumed a collision kills both frames).

So you will not get her acknowledgement. This is how you know you suffered a collision, and you need to *backoff*. This WiFi backoff is similar to the end-to-end TCP backoff by halving the congestion window in Chapter 14: you double the **contention window** in WiFi. And then you draw a random number between now and the end time of the contention window. That will be your next chance of sending the lost frame.

This protocol description might sound unmotivated at first. But there are actually two clever ideas of distributed coordination here: randomization and exponential backoff.

First, if stations A and B have their frames collide at one time, you do not want them to backoff to a common time in the future: there will be just another collision. They need to *randomly* backoff to minimize the chance of hitting each other again. This is exactly opposite to aiming at synchronization as in Chapter 8. Of course, it may so happen that they pick exactly the same timeslot again, but that is the price you pay for distributed coordination.

Second, if frames keep colliding, you know the interference condition is very bad, and you, as well as all those stations experiencing persistent collisions of their frames, should start backing off more upon receiving this implicit, negative feedback. Linearly increasing the contention window size is one option, but people thought that would not be aggressive enough. Instead, WiFi mandates *multiplicatively* backing off. Since the multiplicative factor is 2, we call it **binary exponential backoff**. This is similar to the multiplicative decrease of the congestion window size in TCP. As illustrated in Figure 18.5, when your contention window exceeds a *maximum value*, i.e., you have been backing off through too many stages, you should just discard that frame and report the loss to upper layer protocols so that they can try to fix it. The contention window may also have a *minimum value*, in which case a sender has to wait before its first attempt to send a frame.

So far so good. But it is another unpleasant fact of wireless transmission that sensing range is *not* the same as interference range: it might happen that stations A and B collide but they cannot hear each other, as shown in Figure 18.6. This is the famous **hidden node** problem, one of the performance bottlenecks of WiFi hotspots. This problem does not arise in TCP congestion control.

But there is a clever solution using a little explicit message passing this time, to help navigate through this challenging interference problem. It is called **RTS/CTS**. When station A wants to send a frame, it first sends a short control message called Request To Send (RTS). All stations within the sensing range of A receive that message, and each of them in turn sends a short control message called Clear To Send (CTS). All stations within sensing range of them receive the CTS and refrain from transmitting any frames in the near future. Of course station A itself also gets the CTS message back, and when it sees that CTS, it knows all those hidden nodes have also received the CTS and thus will not send any frames now. At that point, station A sends the actual frames.

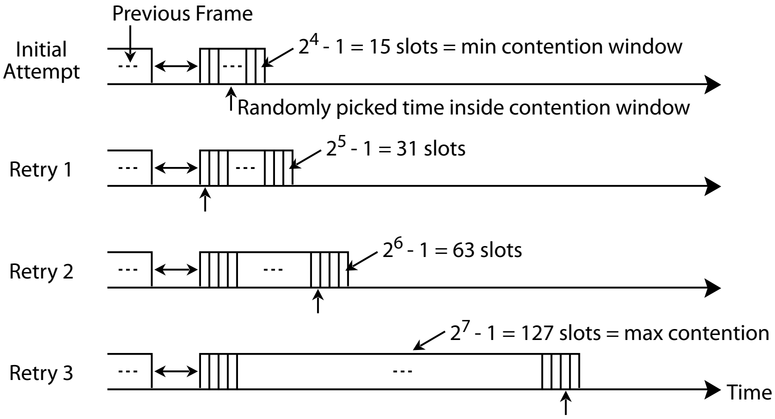

Previous Frame
Initial Attempt
$2^4 - 1 = 15$ slots = min contention window
Randomly picked time inside contention window
Retry 1
$2^5 - 1 = 31$ slots
Retry 2
$2^6 - 1 = 63$ slots
Retry 3
$2^7 - 1 = 127$ slots = max contention
Time

Figure 18.5 The exponential backoff in DCF. There are two key ideas. First, when two frames collide, both need to back off. In order to avoid both picking the same time to retransmit, each picks a random point over the contention window to retransmit. Second, if collisions continue, each sender needs to back off more. Doubling the contention window is a reasonable way to increase the degree of backing off. A homework problem will explore this further. The minimum window size is W_{min}, and the maximum number of backoffs allowed (before the frame is discarded) is B.

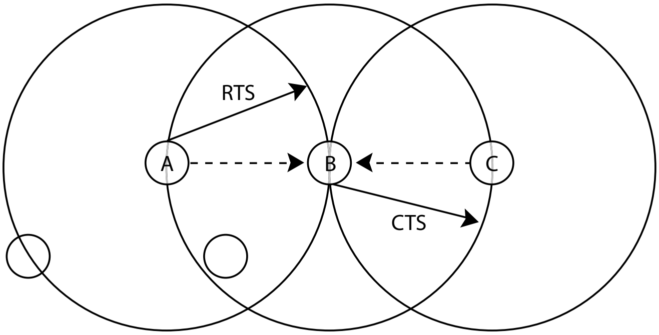


Figure 18.6 The hidden node problem: Stations A and C's transmissions to station B interfere with each other, but cannot sense each other. Dotted lines denote sensing/transmission range. RTS/CTS is a message passing protocol to help resolve the hidden node problem. Node A first sends an RTS. Upon hearing the RTS, all nodes (including node B) send a CTS. Upon hearing the CTS, all nodes (including node C) remain silent for a period of time, except node A itself, which initiated the RTS in the first place. Node A now knows it is safe to send the actual data frames without worrying about hidden nodes.

As Figure 18.7 illustrates, the brief period of idle time in between an RTS and the CTS is shorter than the wait-and-listen time between data transmissions. This is yet another clever idea in distributed coordination in wireless networks. By creating multiple types of wait-and-listen intervals, those transmissions that

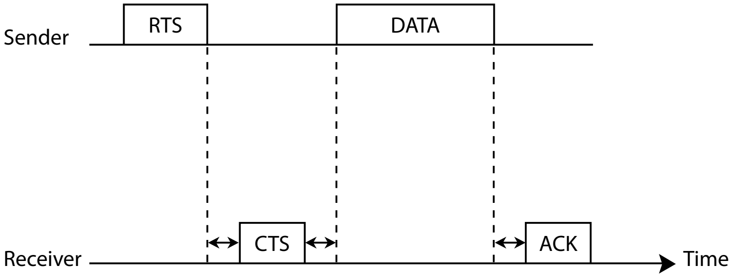


Figure 18.7 A timing diagram of RTS/CTS in WiFi DCF to help mitigate the hidden node problem. The durations of time between RTS and CTS, and between CTS and Data frames, are smaller than the period of time that other nodes need to wait for a clear channel before transmitting. This timing difference effectively provides the priority of CTS and the following data traffic over competing traffic.

only need to observe a shorter wait-and-listen interval are essentially given higher *priority*. They will be allowed to send before those who must observe a longer wait-and-listen period.

RTS/CTS is not a perfect solution either, e.g., RTS and CTS frames themselves may collide with other frames. Still, with the RTS/CTS message passing protocol, together with prioritization through different wait-and-listen intervals, distributed transmission through randomized transmit timing, and contention resolution through an exponentially backed-off content window, we have a quite distributed MAC protocol that enables the operation of WiFi hotspots as they scale up.

We will see several other wireless peculiarities and their effects on both efficiency and fairness in the Advanced Material. But first let us work out the throughput performance of WiFi devices in a hotspot running DCF.

18.2 A Long Answer

Random access offers a complementary approach to power control as an interference management method. To be exact, there is a power control functionality in WiFi too, but it is mostly for conforming to energy limits in the unlicensed band and for saving battery energy, rather than to manage interference. While power control can be analyzed through linear algebra (and some game theory and optimization theory) as presented in Chapter 1, random access involves probabilistic actions by the radios and its performance analysis requires some basic probability theory.

CSMA in WiFi DCF is not particularly easy to model either, because frame collisions depend on the actions of each radio, and the history of binary exponential backoff couples with the transmission decision at each timeslot. A well-known

19 Why am I getting only a few % of the advertised 4G speed?

By the end of this chapter, you will count yourself lucky to get as much as a few percent of the advertised speed. Where did the rest go?

19.1 A Short Answer

First of all, the terms 3G and 4G can be confusing. There is one track following the standardization body 3GPP called **UMTS** or **WCDMA**, and another track in 3GPP2 called **CDMA2000**. Each also has several versions inbetween 2G and 3G, often called 2.5G, such as EDGE, EVDO, etc. For 4G, the main track is called Long Term Evolution (**LTE**), with variants such as LTE light and LTE advanced. Another competing track is called WiMAX. Some refer to evolved versions of 3G, such as HSPA+, as 4G too. All these have created quite a bit of confusion in a consumer's mind as to what really is a 3G technology and what really is a 4G technology.

You might have read that the 3G downlink speed for stationary users should be 7.2 Mbps. But when you try to download an email attachment of 3 MB, it often takes as long as one and half minutes. You get around 267 kbps, 3.7% of what you might expect. Who took away the 96%?

Many countries are moving towards LTE. They use a range of techniques to increase the **spectral efficiency**, defined as the number of bits per second that each Hz of bandwidth can support. These include methods like OFDM and MIMO mentioned at the end of the last chapter and splitting a large cell into smaller ones. But the user observed throughput in 4G, while much higher than that for 3G, still falls short of the advertised numbers we often hear in the neighborhood of 300 Mbps. Why is that?

There are two main reasons: non-ideal network conditions and overheads. Many parts of the wireless network exhibit non-ideal conditions, including both the air-interface and the backhaul network. Furthermore, networks, just like our lives, are dominated by overheads, such as the overhead of network management in the form of control bits in packets or control sequences in protocols.

This chapter is in some sense the "overhead" of this book: there are no further "deep" messages other than the importance of overhead: networking is not just about maximizing performance metrics like throughput, but also involves the inevitable cost of managing the network.

Let us go into a little bit of detail on three major sources of "speed reduction," or more accurately, reduction in **useful throughput** in a session from a sender to the receiver. Useful throughput is defined as the number of bits of actual application data received, divided by the time it takes to get the data through. This is what you "feel" you are getting in your service, but might not be what advertisements talk about or what speed tests measure.

19.1.1 Air-interface

1. *Propagation channel*: Wireless channels suffer from various types of degradation, including **path loss** (the signal strength drops as the distance of propagation increases), **shadowing** (obstruction by objects), and multipath **fading** (each signal bounces off of many objects and is collected at the receiver from multiple paths). A user standing at the cell edge, far away from the base station and blocked by many buildings, will receive a lower rate than will another user standing right under a base station. These factors come into play even if there is only one user in the whole world.
2. *Interference*: There are also many users, and they interfere with each other. As mentioned in Chapter 1, if there are few strong interferers, or if the interferers are weak but there are many of them, the received SIR will be low. At some point, it will be so low that the order of modulation needs to be toned down and the transmission rate reduced so that the receiver can accurately decode. As we saw in Chapter 1, a typical instance of the problem in CDMA networks is the *near far problem.* Even power control cannot completely resolve this problem.

19.1.2 Backhaul network

There can be more than ten links traversed from the base station to the actual destination on the other side of a wireless session of, say, YouTube streaming. The session first goes through the radio access network, then the cellular core network also owned by the cellular provider, then possibly a long distance providers' links, then possibly multiple other ISPs composing the rest of the Internet, and, finally, to Google's data center network.

1. *Links*: Users' traffic competes with the traffic of other users on the links behind the air-interface in the cellular network. As explained in more detail in the next section, many wireless networks actually have most of their links in wireline networks. Congestion happens on these links and the resulting queuing delay reduces throughput. Plus there is also propagation delay simply due to the distance traversed. An increase in delay reduces throughput, since throughput is defined as the number of bits that can be communicated from the source to the destination per second.

2. *Nodes*: These links are connected through nodes of various kinds: gateways, switches, routers, servers, etc. Some of these, such as routers, store packets while waiting for the egress links to be ready, thus increasing packet delay. Others, such as servers, have processing-power limitations, and can become heavily congested when they are in popular demand. For example, a popular web server or video server may become so congested that it cannot process all the requests. This has nothing to do with the rest of the network, just a server that cannot handle the demand. Yet it does reduce the throughput for the session.

19.1.3 Protocols

1. *Protocol semantics*: Many functionalities require sequences of message passing. For example, in TCP, each session needs to be set up and torn down, through a three-way handshake and a four-way tear-down, respectively. This process is illustrated in Figure 19.1. Why does the network protocol designer bother to create such a complicated procedure just for setting up and terminating a session? Well, because in this way, for session establishment, both the sender and the receiver know that there is a session and that the other knows it too. And for session tear-down, four-way handshake ensures there is no dangling state of connection in either direction of a full-duplex connection (i.e., a bidirectional path where both ways can be carried out at the same

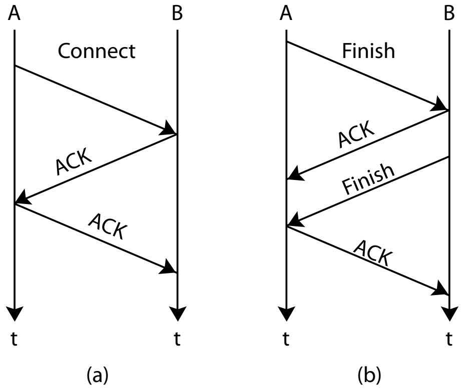


Figure 19.1 (a) Three-way session establishment and (b) four-way session tear-down in TCP. (a) When A initiates a connection with B, B sends an acknowledgement, and A acknowledges the acknowledgement, so that B knows that A knows there is now a connection established. (b) When A initiates a session tear-down, B first acknowledges that. Then B sends a tear-down message right afterwards, since TCP connections are bidirectional: A having no more messages for B does not mean B has no more messages for A. A has to hear that from B.

time). Obviously, for shorter sessions, these overheads occupy a larger fraction of the capacity used, leaving less for the application data.

2. *Packet headers*: As explained in Chapter 13, each layer adds a header to carry control information, such as address, protocol version number, quality of service, error check, etc. These headers also leave some space for flexible future use too. These headers add up, especially if the packet payload is small and the fraction of header becomes correspondingly larger. Some protocols also specify a packet-fragmentation threshold, so bigger packets are divided into smaller ones, adding to the fraction of header overhead.
3. *Control plane signaling*: Think about an air transportation network. The actual traffic of people and cargo is carried by airplanes flying between airports following particular routes. But the routing decision and many other control signals traverse entirely different networks, possibly the Internet or the telephone network. The data plane is separated from the control plane. On the Internet, the actual data traffic flows on **data channels** (a logical concept, rather than physical channels), while control signals travel on **control channels**. Control signals may have to travel half of the world even when the source and destination nodes are right next to each other. These signaling channels take portions of the available data rate and reserve them for control purposes. In 3G and 4G standards, a great deal of effort is put into designing control channels. Sometimes they are sized too small, causing extra delay and reducing throughput. Other times they are sized too big, taking up unnecessary amounts of the overall capacity and hurting throughput too.

In general, there are five main functionalities of network management.

- *Performance*: Monitor, collect, and analyze performance metrics.
- *Configuration*: Update configuration of the control knobs in different protocols.
- *Charging*: Maintain the data needed to identify how to charge each user, e.g., when a user uses the network in time-dependent pricing.
- *Fault-management*: Monitor to see whether any link or node is down, and then contain, repair, and root-cause diagnose the fault.
- *Security*: Run authentication, maintain integrity, and check confidentiality.

The messages of these functionalities sometimes run on channels shared with the actual data (in-band control), and sometimes run on dedicated control channels (out-of-band control). Collectively, they form the control plane. Protocols running network management include examples like Simple Network Management Protocol (**SNMP**) for the Internet.

19.2 A Long Answer

The speed of your wireless (or wireline) Internet connection is not one number, but *many* numbers depending on the answers to the following four questions.

20 Is it fair that my neighbor's iPad downloads faster?

We have come to the last chapter, on a sensitive subject that we touched upon many times in the previous chapters and forms an essential part of both social choice theory and technology network design: quantifying fairness of resource allocation. This may sound obvious, but it does not hurt to highlight: the scope of our discussion will be only on performance metrics, not on liberty and rights.

20.1 A Short Answer

20.1.1 Thinking about fairness

The naive view of "equality is fairness" is problematic in examining performance metrics of a group of users stemming from some allocation of resources. If you have to choose from an allocation of (1, 1) Mbps between two iPad users, and an allocation of (100, 101) Mbps, many people would choose (100, 101) Mbps even though it deviates from an equal allocation. Magnitude matters. Part of Rawls' theory of justice is the difference principle that we will discuss in the Advanced Material, which prefers a less equal allocation if that means everyone gets more. Of course, a more challenging choice would have been between (1, 1) Mbps and (1, 2) Mbps.

Another objection to marking equal allocations as the most fair stems from the differences in the contributions by, and the needs of, different users. If a user in a social network glues the entire network together, her contribution is higher than that of a "leaf node" user. If one works twice as hard or twice as effectively as another, these two people should not receive identical salaries. If instead of assigning one A+ and some D grades, a professor assigns a B grade to all students no matter their performance, that will neither be providing the right incentive for learning nor be deemed fair by many students.

And yet most people would also agree that a more lazy or less capable worker does not deserve to starve to death simply because she works slower. There are some basic allocations that should be provided to everyone. The debate surrounds the definition of "basic:" bread and water, or an annual vacation to the Bahamas (assuming that the latter is at all feasible)? Different notions of fairness define what is "basic" differently.

Throughout this chapter, we will examine approaches for discussing these views of fairness using less ambiguous languages.

20.1.2 Fairness measures from axioms

Given a vector $\mathbf{x} \in \mathcal{R}_{+}^{n}$, where x_i is the resource allocated to user i, *how fair* is it? This question is a special case of the general questions on fairness we saw.

Consider two feasible allocations, $\mathbf{x}$ and $\mathbf{y}$, of iPad download speeds among three users: $\mathbf{x} = [1\ 2\ 3]$ Mbps and $\mathbf{y} = [1\ 10\ 100]$ Mbps. (Since we will not be multiplying these vectors by matrices in this chapter, we skip the transpose notation here.) Among the large variety of choices we have in quantifying fairness, we can get many different fairness values, such as 0.33 for $\mathbf{x}$ and 0.01 for $\mathbf{y}$, or 0.86 for $\mathbf{x}$ and 0.41 for $\mathbf{y}$. That means $\mathbf{x}$ is viewed as 33 times more fair than $\mathbf{y}$, or just twice as fair as $\mathbf{y}$.

How many such "viewpoints" are there? What would *disqualify* a quantitative metric of fairness? Can they all be constructed from a set of axioms: simple statements taken as true for the sake of subsequent inference?

One existing approach to quantifying fairness of $\mathbf{x}$ is through a function f that maps $\mathbf{x}$ into a real number. These fairness measures are sometimes referred to as **diversity indices** in statistics. They range from simple ones, e.g., the ratio between the smallest and the largest entries of $\mathbf{x}$, to more sophisticated functions, e.g., Jain's index and the entropy function. Some of these fairness measures map $\mathbf{x}$ to a normalized range between 0 and 1, where 0 denotes the minimum fairness, 1 denotes the maximum fairness, and a larger value indicates more fairness. How are these fairness measures related? Is one measure "better" than any other? What other measures of fairness may be useful?

An alternative approach is the optimization-theoretic approach of α-fairness and the associated utility maximization problem. Given a set of feasible allocations, a maximizer of the α-fair (or isoelastic) utility function satisfies the definition of α-fairness. We have seen two well-known examples in the previous chapters: a maximizer of the log utility function ($\alpha = 1$) is proportionally fair, and a maximizer of the α-fair utility function as $\alpha \to \infty$ is max-min fair. It is often believed that $\alpha \to \infty$ is more fair than $\alpha = 1$, which is in turn more fair than $\alpha = 0$. But it remains unclear what it means to say, for example, that $\alpha = 2.5$ is more fair than $\alpha = 2.4$.

Clearly, these two approaches for quantifying fairness are different. One difference is the treatment of the efficiency, or magnitude, of resources. On the one hand, α-fair utility maximization results in Pareto optimal resource allocations. On the other hand, scale-invariant fairness measures (ones that map $\mathbf{x}$ to the same fairness value as a normalized $\mathbf{x}$) are unaffected by the magnitude of $\mathbf{x}$, and $[1,\ 1]$ is as fair as $[100,\ 100]$. Can the two approaches be unified?

To address the above questions, we discuss an *axiomatic* approach to fairness measures. There is a set of five axioms, each of which is simple and intuitive, thus being accepted as true for the sake of subsequent inference, like what we saw in Chapter 6. They lead to a useful family of fairness measures. As explained in the Advanced Material, we have the axioms of continuity, of homogeneity, of saturation, of starvation, and of partition. Starting with these five axioms, we can *generate* fairness measures. We derive a unique family of **fairness functions** f_β that includes many known ones as special cases and reveals new fairness measures corresponding to other ranges of β. Then we will remove one of the axioms and discover a more general class of fairness functions $F_{\beta,\lambda}$, with a new parameter λ capturing the relative weight put on fairness vs. efficiency.

While we start with the approach of the fairness measure rather than the optimization objective function, it turns out that the latter approach can also be recovered from f_β. For $\beta \geq 0$, α-fair utility functions can be factorized as the product of two components: (1) our fairness measure with $\beta = \alpha$, and (2) a function of the total throughput that captures the scale, or efficiency, of $\mathbf{x}$. Such a factorization quantifies a tradeoff between fairness and efficiency, addressing questions like "what is the maximum weight that can be given to fairness while still maintaining Pareto efficiency?" It also facilitates an unambiguous understanding of what it means to say that a larger α is "more fair" for general $\alpha \in [0, \infty)$.

20.2 A Long Answer

20.2.1 Constructing the fairness function

What does the fairness function look like? We are again condensing a vector into a scalar, a task we faced in rating averages in Chapter 5 and in opinion aggregation in Chapter 6. We first present a unified representation of the fairness measures constructed from five axioms (in the Advanced Material). It is also provably the *only* family of fairness measures that can satisfy all the axioms. It is a family of functions parameterized by a real number β:

$$f_\beta(\mathbf{x}) = \text{sign}(1-\beta) \cdot \left[\sum_{i=1}^{n} \left(\frac{x_i}{\sum_j x_j} \right)^{1-\beta} \right]^{1/\beta}. \tag{20.1}$$

With the normalization term $\sum_j x_j$, we see that only the distribution matters, not the magnitude of $\mathbf{x}$. This is due to one of the axioms, the Axiom of Homogeneity that says the fairness function f should be a "homogeneous function" where scaling of the arguments does not matter. In the rest of this section, we will show that this unified representation leads to many implications.

We first summarize the special cases in Table 20.1, where β sweeps from $-\infty$ to ∞, and $H(\cdot)$ denotes the entropy function:

$$H(\mathbf{x}) = -\sum_i x_i \log x_i.$$

For some values of β, known approaches to measure fairness are recovered, e.g., Jain's index frequently used in networking research:

Index

achievable, 9
adjacency matrix, 160
adoption percentage, 139
Aloha, 433
alternating projection, 85
Amazon, 62, 361
anchor foreign agent, 447
Android, 1
AP: Access Point, 409
Apple, 239, 361
AppNexus, 40
ARP: Address Resolution Protocol, 450
ARPANET, 279
Arrow's impossibility theorem, 116
associative mixing, 182
AT&T, 236
ATM: Asynchronous Transmission Mode, 312
attraction region, 140
auction, 26
  ascending price, 27
  descending price, 27
  first price, 28
  generalized second price, 28
  independent valuation, 29
  private valuation, 29
  sealed envelope, 27
  second price, 28
  valuation, 26
  Vickrey–Clarke–Groves, 28
autonomous system, 284
axiom, 116, 456

bargaining, 121, 466
base station, 1
baseline predictor, 69
basic reproduction number, 171
Bass model, 148
Bayesian, 96, 98, 136, 420
belief network, 109
Bellman equation, 289
Bellman–Ford algorithm, 288
best effort, 280
bidirectional link, 160
binary exponential backoff, 416
bit rate, 388
BitTorrent, 337
boosting, 104, 429
  Ada boost, 104
Braess' paradox, 256
bridge, 308
broadcast, 449
BSS: Basic Service Set, 411
butterfly network, 380

capture, 413
  double, 413
carrier sensing, 415
CDMA2000, 436
CDMA: Code Division Multiple Access, 3
cellular, 236
  3G, 436
  4G, 436
  architecture, 3
  cells, 1
  eNB, 445
  MSC: Mobile Switching Center, 445
  network, 1
centrality, 161
  betweenness, 162
  closeness, 162
  eigenvector, 162
  information, 192
channel gain, 6
CIDR: Classless InterDomain Routing, 286
Cinematch, 65
circuit-switched, 278
Cisco, 219, 239
clickthrough rate, 26
client–server, 340
Clos network, 367
cloud, 361
  advantages, 364
  disadvantages, 363
  provider, 362
  service provider, 362
  user, 362
co-visitation count, 130

collaborative filtering, 62
Comcast, 384
community detection, 185
complementary slackness, 330
compression, 388
  lossless, 388
  lossy, 388
  MP3, 389
  MPEG1, 389
  MPEG2, 389
  MPEG4, 389
conductance, 180
congestion control, 311
  congestion avoidance, 313
  congestion window, 312
  slow start, 313
connected triple, 197
connection-oriented, 311
content distribution network, 400
content-based filtering, 62
contention window, 416
contraction mapping, 147
control channel, 439
convergence, 5, 50, 85, 180, 298, 321
convex function, 72
convex optimization, 72
convex set, 72
coordination, 12
cosine coefficient, 75
coupled oscillators, 144
cross validation, 83
CSMA: Carrier Sensing Multiple Access, 415
cube network, 380
cut, 185, 371

data applications, 1
data center, 361, 370
data channel, 439
data plan, 236
DCF: Distributed Coordination Function, 415
decomposition, 320
degree, 161
DeGroot model, 178
delay sensitivity, 259
demand elasticity, 243
demand function, 243
DHCP: Dynamic Host Configuration Protocol, 286
diameter, 162
difference principle, 467
diffusion models, 132
diminishing marginal returns, 244
directed acyclic graph, 109
disagreement point, 121
distributed algorithm, 5
distributed power control, 5
diversity gain, 419
diversity indices, 456
domain, 284
downlink, 4
duality
  dual decomposition, 327
  duality gap, 329
  Lagrange dual function, 328
  Lagrange dual problem, 327, 352
  Lagrange multiplier, 327
  Lagrangian, 327
  strong duality, 329
  weak duality, 329
Dunbar's number, 161
dynamic programming, 288

eBay, 33
economy of scale, 364
egalitarianism, 467
eigenvalue
  largest, 19
  second largest, 52, 179
emperor's new clothes effect, 137
EPC: Evolvable Packet Core, 445
epidemic, 172
equilibrium
  algorithmic, 8
  game, 12
  Nash, 12
  stable, 140
  unstable, 140
Erdos–Renyi model, 197
Erlang formula, 304
Ethernet, 412
Eulerian cycle, 213
externality, 129
  negative, 4, 31, 120, 245, 428

Facebook, 158
fading, 437
fairenss function, 457
fairness, 243
  α-fair, 243
  α-fair utility function, 243
  cake-cutting, 466
  max-min, 244
  proportional, 244
fat tree, 368
FDMA: Frequency Division Multiple Access, 3
feedback, 5, 260
  implicit, 68
  negative, 416
  positive, 139
flashy whitespace, 260
flexible logistic growth, 151
Ford–Fulkerson algorithm, 306
foreign agent, 447

frame, 283
frequency reuse factor, 3

Gale–Shapley algorithm, 357
game, 10
  best response strategy, 11
  BitTorrent, 358
  cooperative game theory, 121
  cost function, 11
  dominant strategy, 11
  game theory, 10
  mixed strategy, 13
  payoff function, 11
  players, 11
  private torrent, 359
  strategy space, 11
  ultimate, 466
  zero-sum, 24
Garg–Konemann, 354
generalized processor sharing, 405
generalized Watts–Strogatz model, 208
generative model, 194
geometric programming, 88
Google, 25, 44, 361
  matrix, 50
Google+, 161
graph, 160
  k-clique, 181
  k-club, 181
  k-component, 181
  bipartite, 32
  bisection, 185, 367
  bisection bandwidth, 366
  Cayley, 380
  clique, 181
  closed, 178
  cluster, 167
  clustering coefficient, 196
  coloring, 235
  component, 181
  connected component, 181
  diameter, 197
  directed, 160
  full mesh, 181
  hyperedge, 192
  hypergraph, 192
  modularity, 182
  partition, 185
  random, 197
  regular, 198
  strongly connected component, 181
  subgraph, 180
  transitivity, 195
  undirected, 160
greedy social search, 202
group of pictures, 390
  B frame, 390
  I frame, 390
  P frame, 390
Groupon, 131

H.323, 392
hard state, 449
HBO, 62
header, 283
herd immunity, 177
Hessian, 73
hidden node, 417
hierarchical clustering, 188
highly optimized tolerance, 230
home agent, 446
home location register, 447
homogeneous network, 216
homophily, 196
Hulu, 62, 383

IEEE 802.11, 410
IGMP: Internet Group Management Protocol, 391
imitation factor, 148
importance score, 46
in-degree, 45
incidence matrix, 160
incoming links, 45
independence of irrelevant alternatives, 117
  intensity, 118
indirect forwarding, 447
influence function, 139
information cascade, 132
inhomogeneous network, 216
inner-loop power control, 21
interconnection network, 366
interference, 4
Internet, 1, 278
intrinsic factor, 148
IP address, 285
iPhone, 1
IPTV, 386
isoelastic, 244
ISP, 238, 257
  customer–provider, 284
  peering, 284
  tier-1, 284
iterative algorithm, 5, 47, 320, 353

Juniper, 219

Kernighan–Lin algorithm, 185
Kleinberg model, 202

LAN: Local Area Network, 308
layered protocol stack, 281
leaky bucket, 400
least squares, 68
  overfitting, 71
  regularized, 82
lightpath, 234

linear programming, 10
linear regression, 68
linear stationary iteration, 54
link betweenness, 166
log-normal distribution, 232
logistic growth, 170
long-tail distribution, 216
LTE: Long Term Evolution, 436

majorization, 458
Markov chain, 49
matching, 37
max-flow min-cut, 373
mechanism design, 24
Medici, 159
medium access control, 414
message, 283
Metcalfe's law, 338
Microsoft, 361
MIMO: Multiple Input Multiple Output, 429
mobile station, 1
mobility, 446
modularization, 281
MPLS: MultiProtocol Label Switching, 286
multi-stage switched network, 366
multicast, 385
multicommodity flow problem, 298
multiplexing gain, 430

Nagle's algorithm, 444
Nash bargaining solution, 124
NAT: Network Address Translation, 286
near-far problem, 4
negative externality, 28
net utility maximization, 243
Netflix, 61, 429
Netflix prize, 64
network architecture, 282
network effect, 132, 148
  negative, 111, 244
  positive, 244
network security, 454
network utility maximization, 319, 463
non-blocking, 367
  rearrangeably, 367
non-negative matrix, 19
non-orthogonal allocation, 3
NSFNET, 279

OFDM: Orthogonal Frequency Division Multiplexing, 410, 429
optical networks, 234
optimizability, 301
optimization, 9
  constraint, 8
  feasible solution, 9
  infeasible, 9
  objective, 8
  optimal, 10
orthogonal allocation, 3
orthogonal codes, 4
outer-loop power control, 21
outgoing links, 45
overlay network, 341
oversubscription ratio, 365
Overture, 25

P2P, 337, 370
  capacity, 352
  multicast, 339
  super node, 339
P2P law, 338
package bidding, 43
packet, 283
packet-switched, 278
PageRank, 50
Pandora, 62
Pareto distribution, 217
Pareto dominate, 461
Pareto optimal, 9
path loss, 437
payload, 283
PCF: Point Coordination Function, 415
percolation, 175
Perron–Frobenius theory, 19
pixel, 388
positive feedback, 5
positive semidefinite matrix, 73
power method, 21
power of crowds, 131
power-law distribution, 215
preferential attachment, 222
prefix, 285
pricing, 38
  congestion, 321
  congestion-dependent, 260
  flat-rate, 236
  Paris metro, 270
  smart data, 257
  time-dependent, 258
  two-sided, 272
  usage-based, 236
prisoner's dilemma, 11, 120
propagation delay, 316
protocol, 281
PSTN, 278
public key cryptography, 454
pulse coupling, 144

Qualcomm, 4
query flodding, 355
queueing delay, 316

random walk on graphs, 49
rank-1 matrix, 48
rate–distortion, 388
rating, 89
Rawl's theory of justice, 467

recommendation, 62
  collaborative filtering, 67
  content-based filtering, 67
  latent factor model, 68
  neighborhood model, 67
regularization, 71
relevance score, 46
resource pooling, 280
  anti, 272, 428
respondent driven sampling, 157
return function, 145
reverse engineering, 322
review, 89
root mean squared error, 63, 94
round trip time, 314
routing
  BGP: Border Gateway Protocol, 287
  DAR: Dynamic Alternate Routing, 306
  distance vector, 287
  linked state, 287
  multicast, 302, 341
  OSPF: Open Shortest Path First, 287
  RIP: Routing Information Protocol, 287
RTCP: RTP Control Protocol, 393
RTP: Real-time Transport Protocol, 393
RTS/CTS: Request To Send/Clear To Send, 417
RTSP: Real Time Streaming Protocol, 391
rule of succession, 97

scale-free network, 216
Schur concave, 459
search ads, 25
search engine optimization, 56
searchability, 203
segment, 283
self-organized criticality, 230
sequential decision making, 134
shadowing, 437
shortest path problem, 288
SI model, 169
similarity metric, 68, 75
simultaneous ascending auction, 43
SIP: Session Initiation Protocol, 392
SIR model, 171
SIR: Signal-to-Interference Ratio, 5
SIS model, 170
six degrees of separation, 194
Skype, 337
small world, 194
SNMP: Simple Network Management Protocol, 439
social choice theory, 114
social distance, 194
social search, 195
social welfare maximization, 242
socially optimal, 12
soft state, 449
spectral efficiency, 436
spectrum, 2
spread spectrum, 4
spreading code, 4
SSH: Secure SHell, 454
stability, 331
stable marriage matching, 357
station, 412
statistical multiplexing, 219, 260, 280
stepsize, 321
supply, 254
switch, 308
synchronization, 132

TCP
  CUBIC, 315
  FAST, 315
  Reno, 315
  Tahoe, 312
  Vegas, 315
TCP/IP, 279
TDMA: Time Division Multiple Access, 3
tipping, 132
tit-for-tat, 61, 343
traffic engineering, 298
tragedy of the commons, 244
transitivity, 117
transmit power control, 5
tree, 340
  minimum spanning, 308
  multi-tree, 346
  smallest-price, 354
  spanning, 308
  Steiner, 308, 351
triad closure, 195
truthful bidding, 26
Twitter, 158

UDP: User Datagram Protocol, 392
UMTS: Universal Mobile Telecommunications System, 436
uni-directional link, 160
unicast, 278
unlicensed spectrum, 410
uplink, 4
useful throughput, 437
user surplus, 245
utilitarianism, 468
utility function, 242

Verizon, 236
video over Internet, 386
virtual machine, 375
visitor location register, 447
VoIP, 338
voting, 114
  Borda count, 115
  Condorcet voting, 115

- decisive voter, 119
- Kemeny rule, 114
- plurality voting, 114
- positional voting, 115
- preference profile, 114
- voting outcome, 114

waiting function, 261
Watts–Dodds–Newman model, 203
Watts–Strogatz model, 200
wavelength assignment, 234
WCDMA: Wideband CDMA, 436
weak link, 166
welfare function, 461
Wikipedia, 111
wisdom of crowds, 91

YouTube, 62, 129