Blockchain Basics

A Non-Technical Introduction in 25 Steps

Daniel Drescher

Apress®

BLOCKCHAIN BASICS

A NON-TECHNICAL INTRODUCTION IN 25 STEPS

Daniel Drescher

Apress®

Blockchain Basics: A Non-Technical Introduction in 25 Steps

Daniel Drescher
Frankfurt am Main, Germany

ISBN-13 (pbk): 978-1-4842-2603-2 ISBN-13 (electronic): 978-1-4842-2604-9
DOI 10.1007/978-1-4842-2604-9

Library of Congress Control Number: 2017936232

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Susan McDermott
Development Editor: Laura Berendson
Technical Reviewer: Laurence Kirk
Coordinating Editor: Rita Fernando
Copy Editor: Mary Bearden
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484226032. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Apress Business: The Unbiased Source of Business Information

Apress business books provide essential information and practical advice, each written for practitioners by recognized experts. Busy managers and professionals in all areas of the business world—and at all levels of technical sophistication—look to our books for the actionable ideas and tools they need to solve problems, update and enhance their professional skills, make their work lives easier, and capitalize on opportunity.

Whatever the topic on the business spectrum—entrepreneurship, finance, sales, marketing, management, regulation, information technology, among others—Apress has been praised for providing the objective information and unbiased advice you need to excel in your daily work life. Our authors have no axes to grind; they understand they have one job only—to deliver up-to-date, accurate information simply, concisely, and with deep insight that addresses the real needs of our readers.

It is increasingly hard to find information—whether in the news media, on the Internet, and now all too often in books—that is even-handed and has your best interests at heart. We therefore hope that you enjoy this book, which has been carefully crafted to meet our standards of quality and unbiased coverage.

We are always interested in your feedback or ideas for new titles. Perhaps you'd even like to write a book yourself. Whatever the case, reach out to us at editorial@apress.com and an editor will respond swiftly. Incidentally, at the back of this book, you will find a list of useful related titles. Please visit us at www.apress.com to sign up for newsletters and discounts on future purchases.

The Apress Business Team

Contents

About the Author

Daniel Drescher is an experienced banking professional who has held positions in electronic security trading in several banks. His recent activities have focused on automation, machine learning, and big data in the context of security trading. Among others, Daniel holds a doctorate in econometrics from the Technical University of Berlin and an MSc in software engineering from the University of Oxford.

About the Technical Reviewer

Laurence Kirk who after a successful career writing low latency financial applications for the City of London, was captivated by the potential of distributed ledger technology. He moved to Oxford to study for his master's degree and set up Extropy.io, a consultancy working with start-ups to develop applications on the Ethereum platform. Passionate about distributed technology, he now works as a developer, evangelist, and educator about Ethereum.

Introduction

This introduction answers the most important question that every author has to answer: Why should anyone read this book? Or more specifically: Why should anyone read another book about the blockchain? Continue reading and you will learn why this book was written, what you can expect from this book, what you cannot expect from this book, for whom the book was written, and how the book is structured.

Why Another Book About the Blockchain?

The blockchain has received a lot of attention in the public discussion and in the media. Some enthusiasts claim that the blockchain is the biggest invention since the emergence of the Internet. Hence, a lot of books and articles have been written in the past few years about the blockchain. However, if you want to learn more about how the blockchain works, you may find yourself lost in a universe of books that either quickly skim over the technical details or that discuss the underlying technical concepts at a highly formal level. The former may leave you unsatisfied because they miss to explain the technical details necessary to understand and appreciate the blockchain, while the latter may leave you unsatisfied because they already require the knowledge you want to acquire.

This book fills the gap that exists between purely technical books about the blockchain, on the one hand, and the literature that is mostly concerned with specific applications or discussions about its expected economic impact or visions about its future, on the other hand.

This book was written because a conceptual understanding of the technical foundations of the blockchain is necessary in order to understand specific blockchain applications, evaluate business cases of blockchain startups, or follow the discussion about its expected economic impacts. Without an appreciation of the underlying concepts, it will be impossible to assess the value or the potential impact of the blockchain in general or understand the added value of specific blockchain applications. This book focuses on the underlying concepts of the blockchain since a lack of understanding of a new technology can lead to being carried away with the hype and being disappointed later on because of unrealistic unsubstantiated expectations.

This book teaches the concepts that make up the blockchain in a nontechnical fashion and in a concise and comprehensible way. It addresses the three big questions that arise when being introduced to a new technology: What is it? Why do we need it? How does it work?

What You Cannot Expect from This Book

The book is deliberately agnostic to the application of the blockchain. While cryptocurrencies in general and Bitcoin in particular are prominent applications of the blockchain, this book explains the blockchain as a general technology. This approach has been chosen in order to highlight generic concepts and technical patterns of the blockchain instead of focusing on a specific and narrow application case. Hence, this book is:

- Not a text specifically about Bitcoin or any other cryptocurrency
- Not a text solely about one specific blockchain application
- Not a text about proofing the mathematical foundations of the blockchain
- Not a text about programming a blockchain
- Not a text about the legal consequences and implications of the blockchain
- Not a text about the social, economic, or ethical impacts of the blockchain on our society or humankind in general

However, some of these points are addressed to some extent at appropriate points in this book.

What You Can Expect from This Book

This book explains the technical concepts of the blockchain such as transactions, hash values, cryptography, data structures, peer-to-peer systems, distributed systems, system integrity, and distributed consensus in a nontechnical fashion. The didactical approach of this book is based on four elements:

- Conversational style
- No mathematics and no formulas
- Incremental steps through the problem domain
- Use of metaphors and analogies

Conversational Style

This book is deliberately written in a conversational style. It does not use mathematical or computer science jargon in order to avoid any hurdle for nontechnical readers. However, the book introduces and explains the necessary terminology needed to join the discussion and to understand other publications about the blockchain.

No Mathematics and No Formulas

Major elements of the blockchain such as cryptography and algorithms are based on complex mathematical concepts, which in turn come with their own demanding and sometimes frightening mathematical notation and formulas. However, this book deliberately does not use any mathematical notation or formulas in order to avoid any unnecessary complexity or hurdle for nontechnical readers.

Incremental Steps Through the Problem Domain

The chapters in this book are called *steps* for a good reason. These steps form a learning path that incrementally builds the knowledge about the blockchain. The order of the steps was chosen carefully. They cover the fundamentals of software engineering, explain the terminology, point out the reason why the blockchain is needed, and explain the individual concepts that make up the blockchain as well as their interactions. Calling the individual chapters steps highlights their dependence and their didactical purpose. They form a logical sequence to be followed instead of being chapters that could be read independently.

Use of Metaphors and Analogies

Each step that introduces a new concept starts with a pictorial explanation by referring to a situation from real life. These metaphors serve four major purposes. First, they prepare the reader for introduction to a new technical concept. Second, by connecting a technical concept to an easy-to-understand real-world scenario, the metaphors reduce the mental hurdle to discover a new territory. Third, metaphors allow learning new concepts by similarities and analogies. Finally, metaphors provide rules of thumb for memorizing new concepts.

How This Book Is Organized

This book consists of 25 steps grouped into five major stages that all together form a learning path, which incrementally builds your knowledge of the blockchain. These steps cover some fundamentals of software engineering, explain the required terminology, point out the reasons why the blockchain is needed, explain the individual concepts that make up the blockchain as well as their interactions, consider applications of the blockchain, and mention areas of active development and research.

Stage I: Terminology and Technical Foundations

Steps 1 to 3 explain major concepts of software engineering and set the terminology necessary for understanding the succeeding steps. By the end of Step 3, you will have gained an overview of the fundamental concepts and an appreciation of the big picture in which the blockchain is located.

Stage II: Why the Blockchain Is Needed

Steps 4 to 7 explain why the blockchain is needed, what problem it solves, why solving this problem is important, and what potential the blockchain has. By the end of Step 7, you will have gained a good understanding of the problem domain in which the blockchain is located, the environment in which it provides the most value, and why it is needed in the first place.

Stage III: How the Blockchain Works

The third stage is the centerpiece of this book since it explains how the blockchain works internally. Steps 8 to 21 guide you through 15 distinct technical concepts that all together make up the blockchain. By the end of Step 21, you will have reached an understanding of all the major concepts of the blockchain, how they work in isolation, and how they interact in order to create the big machinery that is called the blockchain.

Stage IV: Limitations and How to Overcome Them

Steps 22 to 23 focus on major limitations of the blockchain, explain their reasons, and sketch possible ways to overcome them. By the end of Step 23, you will understand why the original idea of the blockchain as explained in the previous steps may not be suitable for large-scale commercial applications, what changes were made to overcome these limitations, and how these changes altered the properties of the blockchain.

Stage V: Using the Blockchain, Summary, and Outlook

Steps 24 and 25 consider how the blockchain can be used in real life and what questions should to be addressed when selecting a blockchain application. This stage also points out areas of active research and further development. By the end of Step 25, you will have gained a well-grounded understanding of the blockchain and you will be well prepared to read more advanced texts or to become an active part in the ongoing discussion about the blockchain.

Accompanying Material

The website `www.blockchain-basics.com` offers accompanying material for some of the steps of this book.

STAGE

I

Terminology and Technical Foundations

This stage explains major concepts of software engineering and establishes a way to organize and standardize our communication about technology. This learning stage also introduces the concepts of software architecture and integrity and how they relate to the blockchain. By the end of this stage, you will have gained an understanding of the purpose of the blockchain and its potential.

STEP

1

Thinking in Layers and Aspects

Analyzing systems by separating them into layers and aspects

This step lays the foundation of our learning path through the blockchain by introducing a way to organize and standardize our communication about technology. This step explains how you can analyze a software system and why it is important to consider a software system as a composition of layers. Furthermore, this step illustrates what you can gain from considering different layers in a system and how this approach helps us to understand the blockchain. Finally, this step provides a short introduction to the concept of software integrity and highlights its importance.

D. Drescher, *Blockchain Basics*, DOI 10.1007/978-1-4842-2604-9_1

The Metaphor

Do you have a mobile phone? I would guess yes, as most people now have at least one. How much do you know about the different wireless communication protocols that are used to send and receive data? How much do you know about electromagnetic waves that are the foundation of mobile communication? Well, most of us do not know very much about these details because it is not necessary to know them in order to use a mobile phone and most of us do not have the time to learn about them. We mentally separate the mobile phone into the parts we need to know and the parts that can be ignored or taken for granted.

This approach to technology is not restricted to mobile phones. We use it all the time when we learn how to use a new television set, a computer, a washing machine, and so forth. However, these mental partitions are highly individual since what is considered important and what is not depends on our individual preferences, the specific technology, and our goals and experiences. As a result, your mental partition of a mobile phone may differ from my mental partition of the same mobile phone. This typically leads to problems in communication in particular when I try to explain to you what you should know about a certain mobile phone. Hence, unifying the way of partitioning a system is the key point when teaching and discussing technology. This step explains how to partition or layer a system and hence sets the basis for our communication about the blockchain.

Layers of a Software System

The following two ways of partitioning a system are used throughout this book:

- Application vs. implementation
- Functional vs. nonfunctional aspects

Application vs. Implementation

Mentally separating the user's needs from the technical internals of a system leads to a separation of the application layer from the implementation layer. Everything that belongs to the application layer is concerned with the user's needs (e.g., listening to music, taking photos, or booking hotel rooms). Everything that belongs to the implementation layer is concerned with making these things happen (e.g., converting digital information into acoustic signals, recognizing the color of a pixel in a digital camera, or sending messages over the Internet to a booking system). Elements of the implementation layer are technical by nature and are considered a means to an end.

Functional vs. Nonfunctional Aspects

Distinguishing between what a system does and how it does what it does leads to the separation of functional and nonfunctional aspects. Examples of functional aspects are sending data over a network, playing music, taking photos, and manipulating individual pixels of a picture. Examples of nonfunctional aspects are a beautiful graphical user interface, fast-running software, and an ability to keep user data private and save. Other important nonfunctional aspects of a system are security and integrity. *Integrity* means that a system behaves as intended, and it involves many aspects such as security and correctness.[1] There is a nice way to remember the difference between functional and nonfunctional aspects of a system by referring to grammar usage in the English language: verbs describe actions or what is done, while adverbs describe how an action is done. For example, a person can walk quickly or slowly. In both cases, the action of "walk" is identical but how the action is performed differs. As a rule of thumb, one can say that functional aspects are similar to verbs, while nonfunctional aspects are similar to adverbs.

Considering Two Layers at the Same Time

Identifying functional and nonfunctional aspects as well as separating application and implementation layer can be done at the same time, which leads to a two-dimensional table. Table 1-1 illustrates the result of mentally layering a mobile phone in this way.

Table 1-1. Example of Mentally Layering a Mobile Phone

Layer	Functional Aspects	Nonfunctional Aspects
Application	Taking photos	The graphical user interface looks beautiful
	Making phone calls	Easy to use
	Sending e-mails	Messages are sent fast
	Browsing the Internet	
	Sending chat messages	
Implementation	Saving user data internally	Store data efficiently
	Making a connection to the nearest mobile connector	Saving energy
	Accessing pixels in the digital camera	Maintaining integrity
		Ensure user privacy

[1] Chung, Lawrence, et al. *Non-functional requirements in software engineering*. Vol. 5. New York: Springer Science & Business Media, 2012.

Table 1-1 may explain the visibility (or the lack of it) of specific elements of a system to its users. Functional aspects of the application layer are the most obvious elements of a system, because they serve obvious needs of the users. These elements are typically the ones users learn about. On the other hand, the nonfunctional aspects of the implementation layer are rarely seen as major elements of the system. They are typically taken for granted.

Integrity

Integrity is an important nonfunctional aspect of any software system. It has three major components[2]:

- *Data integrity*: The data used and maintained by the system are complete, correct, and free of contradictions.
- *Behavioral integrity*: The system behaves as intended and it is free of logical errors.
- *Security*: The system is able to restrict access to its data and functionality to authorized users only.

Most of us may take integrity of software systems for granted because most of the time we luckily interact with systems that keep their integrity. This is due to the fact that programmers and software engineers have invested a lot of time and effort into the development of systems to achieve and maintain integrity. As a result, we may be a bit spoiled when it comes to appreciating the work done by software engineers to create systems that maintain a high level of integrity. But our feelings may change as soon as we interact with a system that fails to do so. These are the occasions when you face a loss of data, illogical software behavior, or realize that strangers were able to access your private data. These are the occasions when your mobile phone, your computer, your e-mail software, your word processor, or your spreadsheet calculator make you angry and forget your good manners! On these occasions, we begin to realize that software integrity is a highly valuable commodity. Hence, it should not come as a surprise that software professionals spend a lot of their time working on this seemingly tiny nonfunctional aspect of the implementation layer.

[2] Boritz, J. Efrim. IS practitioners' views on core concepts of information integrity. *International Journal of Accounting Information Systems* 6.4 (2005): 260–279.

Outlook

This step provided an introduction to some general principles of software engineering. In particular, the concepts of integrity and functional vs. nonfunctional aspects as well as application vs. implementation of a software system were illustrated. Understanding these concepts will help you appreciate the wider scope in which the blockchain exists. The next step will present the bigger picture by using the concepts introduced in this step.

Summary

- Systems can be analyzed by separating them into:
  - Application and implementation layer
  - Functional and nonfunctional aspects
- The application layer focuses on the user's needs, while the implementation layer focuses on making things happen.
- Functional aspects focus on what is done, while nonfunctional aspects focus on how things are done.
- Most users are concerned with the functional aspects of the application layer of a system, while nonfunctional aspects of a system, in particular those of the implementation layer, are less visible to users.
- Integrity is an important nonfunctional aspect of any software system and it has three major elements:
  - Data integrity
  - Behavioral integrity
  - Security
- Most software failures, such as losses of data, illogical behavior, or strangers accessing one's private data, are the result of violated system integrity.

STEP

2

Seeing the Big Picture

Software architecture and its relation to the blockchain

This step not only provides the big picture in which the blockchain is located, but it also highlights its location within the big picture. In order to allow you to see the big picture, this step introduces the concept of software architecture and explains its relation to the concept of separating a system into layers and aspects. In order to help you recognize the location of the blockchain within the big picture, this step highlights the relationship between the blockchain and software architecture. Finally, this step points out the core purpose of the blockchain in just one sentence. Appreciating its purpose is a cornerstone in understanding the blockchain and understanding the course of the succeeding steps.

The Metaphor

Have you ever bought a car? Most of us have. Even if you have never bought a car, you probably know that cars are equipped with different types of engines (e.g., diesel, gasoline, or electric engine). This is an example of the process

D. Drescher, *Blockchain Basics*, DOI 10.1007/978-1-4842-2604-9_2

of modularization, which is the result of applying the idea of layering to cars. Having the choice among different engines when buying a car can result in amazing differences in the vehicle. Two cars that look identical from the outside can differ dramatically with respect to the power of their engines and hence have very different driving performance. Additionally, your choice of the engine will have an impact on other characteristics of the car, like its price, its operational costs, the type of fuel consumed, the exhaust system, and the dimensions of the brakes. With this picture in mind, understanding the role of the blockchain within the big picture will be much easier.

A Payment System

Let's apply the concept of layering to a payment system. Table 2-1 shows some of the user's needs as well as some of the nonfunctional aspects of both the application and the implementation layers.

Table 2-1. Aspects and Layers of a Payment System

Layer	Functional Aspects	Nonfunctional Aspects
Application	Deposit money	The graphical user interface looks beautiful
	Withdraw money	Easy to use
	Transfer money	Transfer of money is done fast
	Monitor account balance	System has many participants
Implementation	?	Available 24 hours a day
		Fraud resistant
		Maintaining integrity
		Ensure user privacy

Have you spotted the question mark in that part of the table were you normally see information about the technology used to make the system work? This space was left blank on purpose. It is the place where you decide which "engine" should be used to run your system. The next section will tell you a bit more about the engine equivalent in software systems.

Two Types of Software Architecture

There are many ways to implement software systems. However, one of the fundamental decisions when implementing a system concerns its architecture, the way in which its components are organized and related to one another.

The two major architectural approaches for software systems are centralized and distributed.[1]

In centralized software systems, the components are located around and connected with one central component. In contrast, the components of distributed systems form a network of connected components without having any central element of coordination or control.

Figure 2-1 depicts these two contrary architectures. The circles in the figure represent system components, also called nodes, and the lines represent connections between them. At this point, it is not important to know the details of what these components do and what information is exchanged between the nodes. The important point is the existence of these two different ways of organizing software systems. On the left-hand side of Figure 2-1, a distributed architecture is illustrated where components are connected with one another without having a central element. It is important to see that none of the components is directly connected with all other components. However, all components are connected with one another at least indirectly. The right-hand side of Figure 2-1 illustrates a centralized architecture where each component is connected to one central component. The components are not connected with one another directly. They only have one direct connection to the central component.

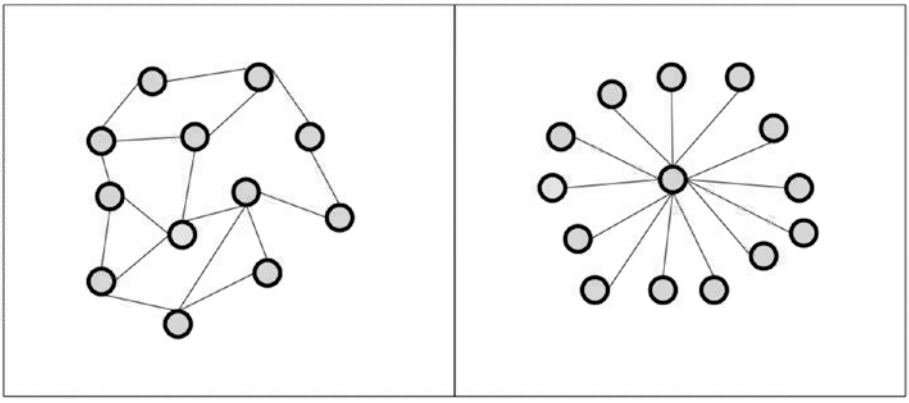

Figure 2-1. Distributed (left) vs. centralized (right) system architecture

[1] Tanenbaum, Andrew S., and Maarten Van Steen. *Distributed systems: principles and paradigms*. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.

The Advantages of Distributed Systems

The major advantages of a distributed system over single computers are[2]:

- Higher computing power
- Cost reduction
- Higher reliability
- Ability to grow naturally

Higher Computing Power

The computing power of a distributed system is the result of combining the computing power of all connected computers. Hence, distributed systems typically have more computing power than each individual computer. This has been proven true even when comparing distributed systems comprised of computers of relatively low computing power with isolated super computers.

Cost Reduction

The price of mainstream computers, memory, disk space, and networking equipment has fallen dramatically during the past 20 years. Since distributed systems consist of many computers, the initial costs of distributed systems are higher than the initial costs of individual computers. However, the costs of creating, maintaining, and operating a super computer are still much higher than the costs of creating, maintaining, and operating a distributed system. This is particularly true since replacing individual computers of a distributed system can be done with no significant overall system impact.

Higher Reliability

The increased reliability of a distributed system is based on the fact that the whole network of computers can continue operating even when individual machines crash. A distributed system does not have a single point of failure. If one element fails, the remaining elements can take over. Hence, a single super computer typically has a lower reliability than a distributed system.

---

[2]Tanenbaum, Andrew S., and Maarten Van. Steen. *Distributed systems: principles and paradigms*. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.

Ability to Grow Naturally

The computing power of a distributed system is the result of the aggregated computing power of its constituents. One can increase the computing power of the whole system by connecting additional computers with the system. As a result, the computing power of the whole system can be increased incrementally on a fine-grained scale. This supports the way in which the demand for computing power increases in many organizations. The incremental growth of distributed systems is in contrast to the growth of the computing power of individual computers. Individual computers provide identical power until they are replaced by a more powerful computer. This results in a discontinuous growth of computing power, which is only rarely appreciated by the consumers of computing services.

The Disadvantages of Distributed Systems

The disadvantages of distributed systems compared to single computers are:

- Coordination overhead
- Communication overhead
- Dependency on networks
- Higher program complexity
- Security issues

Coordination Overhead

Distributed systems do not have central entities that coordinate their members. Hence, the coordination must be done by the members of the system themselves. Coordinating work among coworkers in a distributed system is challenging and costs effort and computing power that cannot be spent on the genuine computing task, hence, the term coordination overhead.

Communication Overhead

Coordination requires communication. Hence, the computers that form a distributed system have to communicate with one another. This requires the existence of a communication protocol and the sending, receiving, and

processing of messages, which in turn costs effort and computing power that cannot be spend on the genuine computing task, hence, the term communication overhead.

Dependencies on Networks

Any kind of communication requires a medium. The medium is responsible for transferring information between the entities communicating with one another. Computers in distributed systems communicate by means of messages passed through a network. Networks have their own challenges and adversities, which in turn impact the communication and coordination among computers that form a distributed system. However, without any network, there will be no distributed system, no communication, and therefore no coordination among the nodes, thus the dependency on networks.

Higher Program Complexity

Solving a computation problem involves writing programs and software. Due to the disadvantages mentioned previously, any software in a distributed system has to solve additional problems such as coordination, communication, and utilizing of networks. This increases the complexity of the software.

Security Issues

Communication over a network means sending and sharing data that are critical for the genuine computing task. However, sending information through a network implies security concerns as untrustworthy entities may misuse the network in order to access and exploit information. Hence, any distributed system has to address security concerns. The less restricted the access to the network over which the distributed nodes communicate is, the higher the security concerns are for the distributed system.

Distributed Peer-to-Peer Systems

Peer-to-peer networks are a special kind of distributed systems. They consist of individual computers (also called nodes), which make their computational resources (e.g., processing power, storage capacity, data or network bandwidth) directly available to all other members of the network without having