ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Department of Computer Science

A Peer-to-Peer Approach to Sharing Wireless Local Area Networks

PhD dissertation

by

Elias C. Efstathiou

Athens

June 2006

Abstract

For the first time in the history of telecommunications, private individuals can provide telecommunication services to their peers. This change was brought on by the emergence of low-cost Wireless Local Area Network (WLAN) technology. WLANs based on the IEEE 802.11 family of specifications are everywhere, from businesses and universities, to hotels and airports. Households with broadband connections also use WLANs for tetherless access to the Internet.

WLANs usually cover greater areas than intended with their installation. However, a client that attempts to access a foreign WLAN is likely to fail: most WLANs are secured against outsiders. These WLANs, along with their backhaul connections, represent an underutilized resource. In metropolitan areas, the density of WLANs is high. The Internet access bandwidth they can offer is greater than what 2G cellular offers—even if the backhaul is a simple DSL connection. Newer cell phones are equipped with WLAN adapters. Thus, the stage is set for an alternative public cellular network, one that relies on numerous WLAN access points owned and managed by private individuals.

Why would selfish individuals cooperate and share their WLANs with others when there are potential direct and indirect costs involved? In this dissertation, we define, design, and evaluate a scheme for sharing WLANs with others that relies on peer-to-peer principles. Our scheme, called *Peer-to-Peer Wireless Network Confederation* (P2PWNC), adopts ideas from peer-to-peer file sharing, and carries them over to WLAN sharing. The main idea is that individuals share their WLANs with foreigners in exchange for future connectivity in foreign WLANs. Thus, our scheme relies on the principle of reciprocity.

P2PWNC encourages cooperation by excluding those who do not contribute from consuming. In addition, it stimulates participation in the scheme by keeping entry costs low: No one oversees a P2PWNC system, and participants who join it do not need to register with authorities. This is similar to how participants join file-sharing networks by simply downloading appropriate software. The problem is that in open decentralized systems, identifying *free-riders* is difficult. Free-riders can join the system using multiple identities, and use these identities to forge evidence of contribution to make an identity *appear* cooperative. To allow this behavior would make true cooperation unsustainable.

In this dissertation, we rely on an algorithm proposed by others for detecting free-riders in open decentralized peer-to-peer environments. We address a specific vulnerability of that algorithm, and adapt it to the specifics of the P2PWNC system. We design a decentralized storage subsystem, based on selfish nodes that run a *gossiping algorithm*. Transaction history in the form of *receipts* is stored in this subsystem, and is used as input to the reciprocity algorithm, which identifies free-riders.

We evaluate the algorithms with simulation-based experiments. We show that the algorithms can sustain cooperation in environments where other *strategies* are present, which compete with our algorithms in both evolutionary and non-evolutionary settings.

We also present the *P2PWNC protocol*, which is designed as a lightweight vendor-neutral protocol that can be implemented directly on WLAN access points and WLAN-enabled cell phones. We then briefly present our prototype implementation.

The simulation-based experiments and the prototype implementation serve to prove that a system like P2PWNC could be viable in the real world. Work on the P2PWNC protocol and implementation that is beyond the scope of this dissertation, continues.

Acknowledgments

My thesis adviser, George Polyzos, has been a constant source of support and ideas during the past four and a half years. Always available, always with a smile, he would help me see the bigger picture in matters academic, professional, and personal. He was instrumental in creating the right environment at the Mobile Multimedia Laboratory that attracted some of the best and brightest people I have ever met—George being one of them. He makes me proud to be his student.

Having had the chance to work close to Costas Courcoubetis and George Stamoulis is an experience I will value, and I thank them. They are true teachers and scientists. Their students, my friends, Panayotis Antoniadis, Manos Dramitinos, and Thanasis Papaioannou, I thank for our long discussions; I took advantage of their patience on numerous occasions.

Our group has grown in four years. I had a great time with Marina Bitsaki, Ioanna Constantiou, Nafsika Kokkini, Sergios Soursos, and Christopher Ververidis; and the next generation: Costas Kalogiros, Lina Kanellopoulou, Ntinos Katsaros, and Stavros Routzounis. I wish them the best.

The P2PWNC project, born out of this work, involves many. I thank Apostolos Malatras and Dimitris Mistriotis, the original collaborators. Costas Saninas stayed with us for some time, and our discussions on security issues dragged on through the night. The current team includes an excellent group of friends and hackers: Fotis Elianos, Vasilis Kemerlis, Dimitris Paraskevaidis, and Lefteris Stefanis.

I thank Ben Strulo, Vasileios Darlagiannis, Nico Liebau, David Hausheer and Jeff Farr from the IST MMAPPS consortium for some of the original stimulating discussions on how to encourage cooperation in peer-to-peer systems. I also thank Kostas Anagnostakis; our discussions on incentive techniques have influenced this work.

I would like to thank Prof. Georgios Doukidis and Prof. Petri Mähönen for agreeing to be members of my thesis defense committee. Dr. George Xylomenos is also a member of this committee and I thank him for many interesting discussions on all aspects of Computer Science and academic life. Prof. Lazaros Merakos, member of the defense committee also, has been my good teacher. He introduced me to Networks, he supervised my BSc thesis,

and he gave me the opportunity to work in the Communication Networks Laboratory at the National and Kapodistrian University of Athens—a most rewarding experience. I also thank Prof. Jon Crowcroft for introducing me to the technologies of the Internet (and to the Internet's technologists at IETF 51), and for supervising my MSc thesis. Dr. John Vlontzos, my boss and guide during my time in industry, gave me the opportunity to work on challenging projects; I thank him for his reference letters when I left for graduate school.

The financial support of the program "Herakleitos—Fellowships for Research at the Athens University of Economics and Business" (co-financed by the Greek Department of Education and the European Union through the program "EPEAEK II") is highly appreciated.

I thank Pantelis Frangoudis for his invaluable help in designing and implementing the P2PWNC protocol. Pantelis likes his algorithms efficient, and he is a good friend.

Last, I thank all my close friends and family.

Table of Contents

Vita and Publications

1999	BSc, honors, Department of Computer Science and Telecommunications, University of Athens
1999 – 2000	Intracom S.A., New Technologies Department
2001	MSc, Department of Computer Science, University College London
2002 – 2003	Intracom S.A., R&D Division (independent contractor)
2006	PhD, Department of Computer Science, Athens University of Economics and Business

Journal Article

- E. C. Efstathiou and G. C. Polyzos, "Self-Organized Peering of Wireless LAN Hotspots," *European Transactions on Telecommunications*, vol. 16, no. 5 (Special Issue on Self-Organization in Mobile Networking), Sept./Oct. 2005.

Conference and Workshop Papers

- E. C. Efstathiou, P. A. Frangoudis, and G. C. Polyzos, "Stimulating Participation in Wireless Community Networks," IEEE INFOCOM 2006, Barcelona, Spain, April 2006.
- G. C. Polyzos, C. N. Ververidis, and E. C. Efstathiou, "Service Discovery and Provision for Autonomic Mobile Computing," 2nd IFIP International Workshop on Autonomic Communication (WAC), Vouliagmeni, Greece, Oct. 2005.
- P. A. Frangoudis, E. C. Efstathiou, and G. C. Polyzos, "Reducing Management Complexity through Pure Exchange Economies: A Prototype System for Next Generation Wireless/Mobile Network Operators," 12th Workshop of the HP Openview University Association (HPOVUA), Porto, Portugal, July 2005.
- E. C. Efstathiou and G. C. Polyzos, "Can Residential Wireless LANs Play a Role in 4G?" 4G Mobile Forum (4GMF) Annual Conference, San Diego, CA, July 2005.
- E. C. Efstathiou and G. C. Polyzos, "A Self-Managed Scheme for Free Citywide Wi-Fi," IEEE WoWMoM Autonomic Communications and Computing Workshop, Taormina, Italy, June 2005.
- E. C. Efstathiou and G. C. Polyzos, "Trustworthy Accounting for Wireless LAN Sharing Communities," European PKI Workshop, Samos Island, Greece, June 2004.
- E. C. Efstathiou and G. C. Polyzos, "A Peer-to-Peer Approach to Wireless LAN Roaming," ACM MOBICOM Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH), San Diego, CA, Sept. 2003.
- C. Ververidis, E. C. Efstathiou, S. Soursos, and G. C. Polyzos, "Context-aware Resource Management for Mobile Servers," 10th Workshop of the HP Openview University Association (HPOVUA), Geneva, Switzerland, July 2003.

- P. Antoniadis, C. Courcoubetis, E. C. Efstathiou, G. C. Polyzos, and B. Strulo, "The Case for Peer-to-Peer Wireless LAN Consortia," 12th IST Summit on Mobile/Wireless Communications, Aveiro, Portugal, June 2003.

- E. C. Efstathiou and G. C. Polyzos, "Multipoint Communications in a Beyond-3G Internetwork," International Workshop on Wired/Wireless Internet Communications (WWIC), Las Vegas, NV, June 2002.

Demo Papers

- E. C. Efstathiou, F. A. Elianos, P. A. Frangoudis, V. P. Kemerlis, D. C. Paraskevaidis, G. C. Polyzos, and E. C. Stefanis, "Practical Incentive Techniques for Wireless Community Networks", 4th International Conference on Mobile Systems, Applications, and Services (MobiSys 2006) Demo Session, Uppsala, Sweden, June 2006.

- E. C. Efstathiou, F. A. Elianos, P. A. Frangoudis, V. P. Kemerlis, D. C. Paraskevaidis, G. C. Polyzos, and E. C. Stefanis, "The Peer-to-Peer Wireless Network Confederation Scheme," IEEE INFOCOM 2006 Demo Session, Barcelona, Spain, April 2006.

- E. C. Efstathiou, F. A. Elianos, P. A. Frangoudis, V. P. Kemerlis, D. C. Paraskevaidis, G. C. Polyzos, and E. C. Stefanis, "The Peer-to-Peer Wireless Network Confederation Scheme: Protocols, Algorithms, and Services," 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2006) Demo Session, Barcelona, Spain, March 2006.

Book Chapters

- E. C. Efstathiou and G. C. Polyzos, "Peer-to-Peer Wireless Network Confederation," in *Encyclopedia of Virtual Communities and Technologies*, S. Dasgupta, ed., Idea Group Reference, 2005.

- E. C. Efstathiou and G. C. Polyzos, "P2PWNC: A Peer-to-Peer Approach to Wireless LAN Roaming," in *Handbook of Wireless Local Area Networks: Applications, Technology, Security, and Standards*, M. Ilyas, S. Ahson, eds., CRC Press, 2005.

- E. C. Efstathiou and G. C. Polyzos, "Mobile Multicast," in *Mobile and Wireless Internet: Protocols, Algorithms and Systems*, K. Makki, N. Pissinou, K. S. Makki, E. K. Park, eds., Kluwer Academic Publishers, 2003.

Poster Papers

- E. C. Efstathiou, F. A. Elianos, P. A. Frangoudis, V. P. Kemerlis, D. C. Paraskevaidis, G. C. Polyzos, and E. C. Stefanis, "Building Secure Media Applications over Wireless Community Networks," 13th Workshop of the HP Openview University Association (HPOVUA), Nice, France, May 2006.

- E. C. Efstathiou, "Self-Organized Peering of Wireless LANs," IEEE INFOCOM 2005 Student Workshop, Miami, FL, March 2005.

- E. C. Efstathiou and G. C. Polyzos, "Designing a Peer-to-Peer Wireless Network Confederation," IEEE LCN Workshop on Wireless Local Networks (WLN), Bonn, Germany, Oct. 2003.

- P. Antoniadis, C. Courcoubetis, E. C. Efstathiou, G. C. Polyzos, and B. Strulo, "Peer-to-Peer Wireless LAN Consortia: Economic Modeling and Architecture," 3rd IEEE International Conference on Peer-to-Peer Computing, Linköping, Sweden, Sept. 2003.

Chapter 1

Introduction

This chapter is a general introduction to the dissertation. Section 1.1 discusses what motivated this research. Section 1.2 presents the dissertation's core thesis. Section 1.3 serves as an introduction to *Wireless LAN* (WLAN) technology and its various uses, focusing on *Wireless Community Networks*. Section 1.4 serves as an introduction to *Peer-to-Peer* (P2P) technology. These two sections discuss systems that are already implemented and deployed. (Chapter 2 presents additional academic work on P2P systems, with emphasis on the problem of encouraging cooperation.) Section 1.5 presents the original contributions of this dissertation, and Section 1.6 presents the structure of the dissertation.

1.1 Motivation

1.1.1 The problem: a peer-to-peer approach to sharing wireless LANs

In April 2002, when work that led to this dissertation begun, two new information and communication technologies were reaching maturity. These were *Peer-to-Peer* (P2P) file sharing and *Wireless Local Area Networks* (WLANs). At that point, both technologies had more than 3 years of history. In late 1999, *Napster* first popularized the P2P sharing of music files over the Internet. In December 1999, the *Recording Industry Association of America* filed a lawsuit against Napster, seeking $20 billion in damages, which translated to $100,000 per copyright-protected song swapped to that date [1]. Also in 1999, the *Institute of Electrical and Electronics Engineers* ratified IEEE 802.11b-1999 [2], a supplement to the original IEEE 802.11 standard for CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) WLANs that operate in unlicensed spectrum. The IEEE 802.11b supplement specified bit rates of up to 11 Mbps at the physical layer.

By 2002, *Kazaa* had replaced Napster as the most popular file-sharing application and file sharing was not limited to music files any more [3]. At the same time, owing to the commercial success of 802.11b-based products, the family of 802.11 specifications had grown. The IEEE task group on 802.11g [4] designed a 54-Mbps physical layer, operating at the same 2.4-GHz spectrum as 802.11b; this made 802.11g backward compatible with 802.11b. The IEEE task group on 802.11e [5] updated the (largely unimplemented) *Point Coordination Function* for contention-free access to the medium of 802.11, introducing

Quality-of-Service at the 802.11 MAC layer. The IEEE task group on 802.11i [6] addressed the published [7] shortcomings of the 802.11 WEP security algorithm, and designed a comprehensive security architecture for 802.11-based networks. Meanwhile, 802.11b was a standard accessory on portable computers, first via 802.11b adapters, and, later, arriving built-in. In addition, 802.11b access points (APs) wirelessly extended Ethernet networks in organizations.

This was the environment in which the question answered in this dissertation was raised: Is it possible to *define*, *design*, and *implement* a "P2P WLAN sharing" scheme? That is, is it possible to borrow ideas from P2P file sharing, adapt them, and carry them over to WLAN sharing? (Note that in the context of this dissertation the term "P2P WLAN" does not mean the P2P, or *ad hoc* [8], mode of infrastructure-less WLAN operation.)

A first approach to defining P2P WLAN sharing [9] relied on the concept of *peering agreements* among *administrative domains*. According to the definition in [9], P2P WLAN sharing was an action taken by domain administrators who automatically allowed visitors, coming from peer administrative domains, to access WLANs in the local domain and, then, to access the Internet through these WLANs. Domain administrators did this in exchange for the promise of future connectivity for their own members when these members visited other domains that participated in the peering scheme.

Even in this first definition, the focus was not on traditional network peering agreements. Rather, it was on the keywords *automatically* and *in exchange*. The term "automatically" referred to practical problems with deployed WLANs: The use of WLANs in organizations is usually restricted to authorized members of the local domain for security reasons. Providing access to visitors, even if local policy allows it, is not straightforward. Visitors have to manage settings for WEP or other common Layer-2 [10] and Layer-3 [11] security schemes. Therefore, accessing foreign WLANs has significant overhead for those who only want to check their email or to browse a website.

More importantly, however, the term "in exchange" highlighted the fact that a valuable resource (WLAN and Internet connectivity) remains underutilized because of the security policies mentioned above. The excess capacity that WLAN and Internet links generally have could be shared in "times of plenty" in exchange for capacity when "in need".

This is also a characteristic of P2P file sharing. In P2P file sharing, underutilized local resources (files, CPU, and bandwidth) at the edge of the Internet are shared, either for the "good of the community" or in exchange for other files (or other parts of the same file).

The notion of network edges with excess capacity contributed to our second and final definition of P2P WLAN sharing: Assume that the aforementioned administrative domain is not an organization but a household with a broadband connection to the Internet. Assume further that this household has a WLAN AP set up, used by household members to access their Internet connection. With proper placement, passersby outside the household can also access such an AP [12]. Given enough households and enough APs in an (urban) area, the coverage offered by these APs could rival the coverage offered by public cellular networks in the same area. In such a P2P WLAN sharing scheme, households would offer connectivity to passersby in exchange for future connectivity for their own members through other APs that participated in the scheme.

These WLAN APs, which are attached to always-on broadband connections, could serve those who wanted to access the Internet when roaming within cities—or any area with an appropriate density of WLANs. With new WLAN-enabled cell phones becoming available [13], such a scenario is plausible. The scenario enables, among others: Internet-based video and audio calls that cost little compared to the cellular alternative; connection speeds that are an order of magnitude higher than 3G cellular; a sense of community independent of commercial carriers; and even reduced exposure to electromagnetic radiation. (The transceivers of current WLAN clients operate at maximum power levels that are an order of magnitude lower than their current cell phone equivalents [14].)

A working prototype demonstrating the above concepts was built in the context of this dissertation [15, 16]. This implementation showed that the technology for implementing a P2P WLAN sharing scheme is readily available and costs little.

1.1.2 October 2002: Negroponte on WLANs

Others also realized the potential of WLANs as a substitute or complement to public cellular networks. In his article on the October 2002 issue of Wired magazine [17], Nicholas Negroponte described a "lily pad" analogy for WLANs. (Negroponte acknowledges that the first use of the analogy goes to Alessandro Ovi, technology adviser to European Commission President Romano Prodi.) The Wired article mentions "micro-operators, millions of which can be woven into a global fabric of broadband connectivity." The same article notes that the speed of 3G cellular cannot compete with the speed of WLANs, and this fact, coupled with the low cost of WLAN equipment, could make WLANs the ideal technology for widespread broadband wireless connectivity. The "lily

pad" analogy communicated the fact that a WLAN AP can cover a small area compared to a 2G or 3G base station. A solution to this problem is to set up numerous WLAN APs.

The same article mentioned a possible "P2P structure" and the associated "loose agreements" among peers that would help in solving the coverage problem. A P2P structure with loose agreements among peers, which would stimulate participation in a P2P scheme for sharing WLANs is the subject of this dissertation.

1.1.3 February 2006: The case of FON

The Wired article from October 2002 is a suitable foreword to this dissertation. A suitable afterword is the February 2006 announcement from the Spanish start-up company FON [18]. FON, 90-days old at the time of the announcement, secured a total of $21 million in investment capital from Google Inc. [19], two venture capital firms, and the founders of Skype (the company whose namesake product is—in 2006—the most popular voice-over-IP software phone). The goal of FON is to recruit households in a global WLAN sharing scheme. According to the details provided by FON so far (June 2006), a P2P sharing structure is part of their strategy: FON micro-operators that share their WLANs with others can enjoy, when they roam, connectivity through other FON APs, for free. FON provides to interested micro-operators a $50 WLAN AP whose firmware is specially configured by FON. At this time (June 2006), several details of the FON P2P sharing scheme, such as its robustness against *free-riders* (see Section 1.4.1), remain unpublished or unspecified.

A contribution of the FON effort is that it sparked discussions on the problem of the *Acceptable Use Policies* (AUPs) dictated by broadband Internet Service Providers (ISPs). AUPs do not usually permit the sharing of broadband connections with anyone outside the subscribing household. WLAN sharing schemes that rely on ISPs for backhaul connectivity will face this problem. In this dissertation, discussion on this problem is deferred to Chapter 8.

1.2 Thesis

1.2.1 Selfishness and free-riders

Humans come equipped with "selfish genes" [20]. This results from the Darwinian selection process that guides the evolution of life. Of course, in the real-world behaviors are complex: live organisms often cooperate, especially if they are kin. Neo-Darwinian theory explains the cooperative strategies that organisms follow in terms of an increase in *gene-level fitness*. According to this theory, all the cooperative and selfish strategies that organisms follow must lead to an increase of the replication probability of the gene that codes for them, compared to the gene-pool average. Otherwise, in an environment of limited resources, the particular gene tends to become extinct and so does the strategy that this gene codes for.

Neo-Darwinism aside, the selfishness and rationality of individuals has long been a standard assumption in the social sciences and in Game Theory [21]. A more formal definition of selfishness and rationality appears in Chapter 3. It is likely that selfishness would constitute a problem in a P2P WLAN sharing scheme. Moreover, if the direct and indirect costs involved in providing WLAN access to passersby are greater compared to the costs of sharing files over the Internet, the problem in WLAN sharing could become even more pronounced than the selfishness problem in P2P file sharing. Note also that in the same way in which active contributors in a file sharing scheme increase the number of distinct files provided, active contributors in a WLAN sharing scheme increase the wireless coverage that the scheme can achieve. In the proposed P2P WLAN sharing scheme, wireless coverage is a public good created by private contributions (under the additional assumption of absence of congestion). The objective of the proposed scheme is to turn this public good into a *club good*, that is, an *excludable* public good, while making it easy for individuals to join the club.

1.2.2 Requirements for a P2P WLAN sharing scheme

Open system *Secure incentive techniques* are needed to stimulate participation in a P2P WLAN sharing scheme and to exclude free-riders from using it. However, in order to maximize the potential coverage, it should be easy to join the scheme.

Enforceable policies that tie consumption to contribution As opposed to file-sharing networks where free-riders can be tolerated [22] because a few altruistic peers can carry the

burden of uploading the most popular files, in a P2P WLAN sharing scheme every free-rider is a potential participant who could have contributed toward increasing the scheme's wireless coverage, and did not do so. A few altruistic peers do not make much difference in a P2P WLAN sharing scheme because the coverage of their own WLANs limits the coverage they can offer. Therefore, one way to provide the right incentives for system growth is to exclude free-riders from using it, or to provide free-riders with service of lower quality. This dissertation shows that a policy that ties consumption to contribution is enforceable; this policy dictates that the users who rely on the scheme are the ones who must also contribute toward increasing its total wireless coverage. Free-riders and peers that contribute much less than they consume can be detected (by other peers) and punished accordingly. One punishment is to deny them WLAN access or to provide them WLAN service of lower quality. There is a simple approach to achieving the second, discussed in Chapter 6.

No tamperproof modules (software/hardware) Any secure incentive technique should assume that free-riders will tamper with hardware and software and will fake evidence of contribution in an attempt to cheat and then enjoy the benefits of the sharing scheme without contributing to its growth—as long as the net benefit of cheating is greater than the net benefit of contributing.

Free identities A sharing scheme should expose as little information as possible about its participants, preferably by relying on *free identities*. Free identities are system identities that can be created by the participants themselves without any communication with *identity-certifying authorities*. A *free-identity regime* can provide incentives for system growth; however, the free-identity regime must not compromise the effectiveness of the incentive techniques.

1.2.3 The proposed P2P WLAN sharing scheme

In urban areas, public infrastructures for high-speed wireless networking can be built through the private contributions of individual micro-operators who use their Internet-connected WLANs to forward foreign traffic from and to nearby low-mobility clients. The Peer-to-Peer Wireless Network Confederation (P2PWNC) scheme is a P2P WLAN sharing scheme that: (1) assumes that micro-operators are selfish and uses a secure incentive technique to encourage their contribution by tying consumption to contribution; (2) protects

the real-world identities of clients and micro-operators by relying only on disposable opaque identifiers; (3) is fully distributed, open to all, and does not rely on an authority to resolve disputes or to control membership; (4) is automated, using standard hardware (PCs, WLAN APs, WLAN-enabled cell phones) and custom software for the main available platforms (Windows-based PCs, Linux-based WLAN APs, Windows Mobile-based cell phones and PDAs).

1.3 IEEE 802.11 WLANs

1.3.1 General

WLANs are now synonymous with the IEEE 802.11 family of specifications. Products belonging to this family enjoy wide commercial success, which makes 802.11 one of the most successful global wireless standards. Unlike the various incompatible standards of the 2G/3G cellular world, 802.11-compatible products enjoy global acceptance, which guarantees that an 802.11 client will work practically anywhere there is an active 802.11 access point nearby.

With respect to its global acceptance, IEEE 802.11 is similar to *Bluetooth* [23], the popular cable-replacement standard, also used in local-area wireless networks. Even though the price of Bluetooth has reached a level that allows chipsets to be included even in cheap cell phone headsets, the relatively low bit rate of Bluetooth (currently 2.1 Mbps with Bluetooth version 2.0) and its coverage radius of approximately 10 meters (100 meters maximum) make it inappropriate for the applications that 802.11 was designed to support.

The *Institute of Electrical and Electronics Engineers* (IEEE) ratified the initial version of the 802.11 standard in 1997 [24]. Belonging to the same family as Ethernet, it was called “Wireless Ethernet” and, at first, it was considered an appropriate networking technology for offices because it allowed for cable-free operation. 802.11-based products became a commercial success. The cost of manufacturing 802.11 chipsets fell and 802.11 found its way from desktop PCs, to portables, to PDAs, and to cell phones. *Wi-Fi* (originally short for Wireless Fidelity, although today assumed to be meaningless), a consumer-friendly moniker for 802.11 was later adopted and, in 1999, the *Wireless Ethernet Compatibility Alliance* (WECA), a nonprofit international association was formed. WECA’s goal was to certify the interoperability of Wi-Fi products. WECA changed its name to the Wi-Fi Alliance in 2002 [25].

The two main components of 802.11 are the WLAN clients (*stations* in 802.11 terminology) and the WLAN access points (APs). APs are the wireless equivalent of Ethernet hubs. 802.11-enabled clients communicate wirelessly with nearby APs. The APs link the WLAN clients to each other and usually also to the local wired network and the Internet. WLAN clients can also communicate with each other without the help of APs (assuming their radio transceivers are within each other's range) in the *ad hoc* or *peer-to-peer* mode of IEEE 802.11 (not to be confused with our proposed P2P WLAN sharing scheme). Latest IEEE specifications include 802.11g, which enables WLAN clients and APs to connect at speeds of up to 54 Mbps, and 802.11i, which employs authentication and encryption algorithms to protect against unauthorized users that attempt to access private networks. 802.11i also protects the confidentiality and integrity of wireless sessions, which are susceptible to eavesdropping and hijacking attacks.

The term *hotspot* (or "Wi-Fi hotspot") describes an area where an 802.11 AP is available. Public hotspots can be found in airport lounges and shopping malls, in coffee shops and restaurants, and in hotels and conference centers. 802.11 users use their WLAN clients in these hotspots to access their email, their corporate intranets, and the Internet. Users can also use location-based services, and make and receive cheap video- and voice-over-IP calls.

The two characteristics of 802.11 that make it a popular choice for WLANs is that (1) the cost of 802.11 equipment is low ($50 for an AP, $20 for a client adapter—2006 prices) and continues to drop, and (2) 802.11 operates in a spectrum band in which no special license is required from national wireless regulatory commissions (it operates mainly in the 2.4 GHz *Industrial, Scientific, and Medical* band—the ISM band). This means that to deploy an 802.11 WLAN, as long as one adheres to the specified maximum effective radiated power, one is free to do so. Interference in the 2.4 GHz band is, however, substantial and its cause is the existence of numerous 802.11 devices, as well as other devices operating in the same band such as garage-door openers, microwave ovens, cordless phones, and all Bluetooth devices. Technological solutions for limiting the interference between devices in the 2.4 GHz spectrum band have been developed [26]

IEEE 802.11b specifies rates of 11 Mbps at the physical layer. Because of the CSMA/CA protocol overhead, an 802.11b client communicating with an 802.11b AP can, however, achieve rates of approximately 6 Mbps for TCP applications and 7 Mbps for UDP applications [27]. The situation in 802.11g is analogous, with 802.11g specifying a bit rate

of 54 Mbps at the physical layer, with an actual bit rate of 27 Mbps at the IP layer [28]. In addition, because the wireless channel that connects the WLAN clients to the AP is shared among all clients (that is, APs resemble Ethernet hubs more than they do Ethernet switches), this means that, in the presence of other clients that use the same AP, the effective bit rate at the IP layer drops further. In addition, the bit rate drops in order to adapt to signal quality levels.

WLAN APs usually also come equipped with at least one regular Ethernet interface and are able to act as 802.11-to-Ethernet bridges, bridging wireless and wired traffic from wireless clients to the wired network (and to the Internet), and vice versa. Advanced WLAN APs also implement Layer 3 functionality, operating as combination firewall-NAT devices, hosting also the usual DHCP (Dynamic Host Configuration Protocol) functionality, which permits APs attached to one side of a DSL/Cable connection to be used as default IP gateways for the WLAN clients within their coverage radius.

A common problem in mobile networking (that is outside the scope of this dissertation) is the problem of handovers. To ensure uninterrupted Layer 3 flows, mobile WLAN clients must be able to change points of attachment—that is, WLAN APs—seamlessly, as they move away from the coverage area of one AP and enter the coverage area of another. There is still no standard way to achieve this; although the 802.11F recommendation for a vendor-independent inter-AP protocol was ratified in 2003, the IEEE formally withdrew that recommendation in 2005 [29]. Therefore, networks that offer handover functionality to mobile WLAN clients still use proprietary solutions.

The transmission range of an IEEE 802.11 AP, although usually quoted to be 30 to 100 meters indoors, actually depends on the architecture of the building and the material used in its walls and windows. Outdoors, according to published measurements, an AP is usable from several tens of meters away. Special antennas can increase this range to hundreds of meters [12].

Standard cell phone operating systems, such as the *Windows Mobile Pocket PC Phone* and *Smartphone* variants, allow WLAN interfaces to be integrated with cell phones in a standardized way, and this makes WLAN functionality available to users and application developers alike. However, the severe energy drainage problem is still one reason that IEEE 802.11 is not as popular as Bluetooth in consumer devices.

WLAN APs run proprietary embedded operating systems, but other increasingly complex APs use stripped-down versions of the Linux operating system. The *Linksys*

WRT54G AP was the first device whose manufacturing company released the firmware's source code on the Internet in order to comply with the *GNU Public License* (GPL) because GPL software was being used as part of its firmware [30]. The publication of this code sparked a wide-array of Linux-based operating systems that were adapted for use on WLAN APs. A common Linux distribution, also adopted in this work, is the *OpenWrt* distribution [31], a modular distribution with a rich library of add-on modules, support for Quality-of-Service at the IP layer, and additional security functionality at the link layer.

1.3.2 Wireless Community Networks

The fact that IEEE 802.11 technology is cheap and requires no special licenses to operate also gave rise to *Wireless Community Networks* (WCNs). These are metropolitan-area networks whose nodes are owned and managed by volunteers. One of the original WCNs was *Seattle Wireless* [32]. In the original WCNs, the backbone nodes used IEEE 802.11 point-to-point links to connect to each other, creating a (static) multi-hop network completely independent from wired carriers. The WCN nodes provide content and services to each other over this wireless infrastructure, which can span an entire city. The *Athens Wireless Metropolitan Network* (AWMN—Athens, Greece) is one of the largest WCNs worldwide with more than a thousand active nodes [33].

One distinctive characteristic of WCNs is that they value independence from authorities in general, and from wired carriers in particular. After a point, however, and helped by the worldwide drop in DSL/Cable charges, WCNs usually abandon their original wireless backbone and rely on cheap DSL/Cable connections for backhaul.

There are examples of WCNs that transformed into commercial operations, such as *The Cloud* [34] (originally in London, now expanded throughout the UK), which now offers public hotspot services for a fee, either directly to individuals, or in partnership with other businesses, which are allowed to resell access to The Cloud's network of hotspots.

Well known WCNs include *NYCwireless* [35], a non-profit WCN that is dedicated to providing free Internet access in New York City and surrounding areas through its network of hotspots. Similarly, *BAWUG* [36] (The Bay Area Wireless Users Group) provides free public Internet to the greater San Francisco Bay Area. In addition, *NoCat*, the WCN that contributed to the open source community the NoCatAuth [11] authentication software for hotspots, was originally a small WCN offering free public Internet to Sonoma

County, CA. NoCat is now more well known as an open-source software community that offers free software to be used by other WCNs.

In general, the WCN movement relies on volunteers and their altruism. Volunteers devote time and effort to set up and manage a network of hotspots for the good of the community. WCNs are successful in specific areas; however, access to the Internet through WCN-controlled wireless LANs is far from ubiquitous. In that respect, the P2PWNC scheme that we propose can also be seen as an incentive scheme that could stimulate participation in WCNs by encouraging the users who use the free hotspots to also contribute hotspots to the community.

Finally, there are commercial, centrally managed approaches that attempt to solve the same problem as WCNs and P2PWNC, that is, metropolitan wireless access through wireless LANs. For historical reasons and for easy recognition, these efforts sometimes retain the label of "Wireless Community Networks," although they do not rely on volunteers, and may also charge for their services. One of the most well known centrally managed projects involves the City of Philadelphia: the *Wireless Philadelphia* [37] project aims to cover the city with hotspots and provide an alternative broadband infrastructure for its citizens. Google Inc. is managing a similar project for the City of San Francisco; once completed, Google plans to make the network available for free [38].

1.4 Peer-to-peer systems

A formal definition of peer-to-peer systems appears in Section 4.1. Here, we informally introduce some popular peer-to-peer systems, as well as the problem of free-riding in the context of peer-to-peer file sharing systems.

1.4.1 File-sharing systems

Napster, which we already mentioned, was the software client that first popularized the term "peer-to-peer." Its users used Napster to share music files, and Napster was the first client that included the necessary search functionality, as well as the managed download/upload functionality, which made sharing easy. In Napster, the peers who ran the client shared a number of music files from their hard disks. The IDs of these files were advertised in a central index server that other peers could consult in order to find the peer who had the music file they were interested in. This central server was an obvious target when the legal battles of the RIAA begun (see Section 1.1). Although the authorities could

not hunt every peer that was committing copyright infringement by sharing copyrighted music files, they could stop peers from locating each other by shutting down the index server—thus putting an end to widespread sharing. The Napster of today is a commercial online service that legally sells music to subscribers [39].

Gnutella [40] followed in the footsteps of Napster, but with an important difference. The central index server of Napster was decentralized, stored at the peers themselves. This increased the complexity of the software, and the search functionality was both slower and less effective. However, as there was no central index server to shut down, Gnutella and its variants continue to be some of the most popular file-sharing systems.

In the context of Gnutella, the problem of free-riders was formally analyzed [41, 42]. In this context, free-riders are peers who do not share their files but download files from other peers. Their existence has two implications: The number of distinct files available is less than the optimal, and the files that *are* available are replicated and shared by fewer peers, which means that the peers who download files experience more congestion than they would if more peers were sharing.

Kazaa [3] was similar to Gnutella, and was also the first popular client to include an *incentive technique* (see Chapter 3 for a formal definition) that encouraged sharing. The client software in Kazaa monitored the amount of bytes a peer had uploaded and downloaded. The downloaders would report this value to their uploaders, and the uploaders would give priority (in times of congestion) to those who upload more than the others. Obviously, this solution relied on the client software being trustworthy. If a peer could tamper with the software, he could make the software declare anything to the uploader. This is indeed what happened with the popular *Kazaa Hack* variant [43]. Peers who used Kazaa Hack in conjunction with the original software would appear as good uploaders to other peers without really uploading any files. Although this variant reduced the level of cooperation in the Kazaa peer-to-peer network, Kazaa remains a popular file-sharing system.

BitTorrent [44] is currently a popular file-sharing client that relies on a specific incentive technique [45]. Peers download files in pieces from other peers, but at the same time they may upload other pieces of the same file to their uploaders. This one-to-one relationship allows uploaders to tune their upload rate and contribute more bandwidth to the peers who contribute more bandwidth to them in return.

1.4.2 CPU-sharing systems

Peer-to-peer systems also include CPU-sharing systems. A popular CPU-sharing application is *SETI@home* [46]. Peers who download the SETI@home client run a screensaver (when their computer is idle) that processes data downloaded from the Search for Extra-Terrestrial Intelligence project, searching for specific patterns in radio images that the SETI project is interested in. The SETI project thus takes advantage of many idle CPU cycles across the Internet, effectively building a distributed supercomputer. *Folding@home* [47] is another distributed computing project that relies on the idle CPU cycles of individual PCs. Folding@home studies protein folding.

These two CPU-sharing projects can be seen as specialized applications of *Grid* technology [48]. Grids are peer-to-peer systems for sharing processing power. They aim to replace supercomputers with distributed networks of low-cost computers such as PCs, and use them to solve complex problems that require more computing power than the power a supercomputer provides at the same cost.

1.4.3 Distributed Hash Tables

Another popular type of peer-to-peer system is the structured peer-to-peer overlay, also known as *Distributed Hash Table* (DHT). Some of the more well-known DHTs are *Chord* [49], *Pastry* [50], *Tapestry* [51], *Kademlia* [52], *CAN* [53], and *P-Grid* [54]. Although different in their underlying *routing primitives*, all DHTs offer the following functionality: Given a *key*, and a dynamic network of peers, the DHT maps the key to a specific peer (possibly more than one peer, for redundancy). It is then up to the application that uses this primitive to potentially store a *value* on the peer that is responsible for the key. This value could be any type of file object. Thus, DHTs provide the abstraction of the hash table and can be used by the peers that participate in the DHT, or by other clients, to store large amounts of information. The distinctive characteristic of DHTs is that they provide short look-up times for the keys, usually O(logN), where N is the number of peers in the DHT. In principle, the information stored in DHTs is guaranteed to persist even if peers join and leave the DHT. Load balancing algorithms copy or move objects to other peers in such a way as to guarantee their persistence irrespective of any one specific peer.

DHTs in general assume that the peers who make up the DHT and store information are cooperative. The designers of DHTs are mainly concerned with scalability issues and achieving key-lookup times that are short.

1.5 Contributions of this work

With this work, we make the following specific original contributions:

- Starting from the observation that WLANs (and their backhaul connections) can be shared, we give a definition and a description of a P2P system for the sharing of WLANs that is based on the principle of reciprocity.

- We design a specific system for the P2P sharing of WLANs, called Peer-to-Peer Wireless Network Confederation (P2PWNC), and we proceed to research the level of cooperation we can achieve under a strict set of requirements. More specifically, we require that the system should not rely on authorities at any stage in its lifetime, not even when new participants join. We assume that all peers are selfish, rational, and, at the same time, sophisticated programmers that can tamper with the system's modules.

- We identify a weakness in an algorithm proposed in the literature for encouraging cooperation in open P2P systems without authorities. We amend the algorithm, adapt it for P2PWNC, and show the improvement compared to the original.

- We design a receipt-based accounting scheme that can record the transaction history of a P2P community and of P2PWNC. For generating the receipts, we design a receipt generation protocol that can limit the amount of unaccounted consumption (due to selfishness) to arbitrarily small values.

- We define a gossiping algorithm, and use it to design a decentralized storage subsystem to store the transaction history. The subsystem does not rely on structured or unstructured overlays. Moreover, the selfish peers that make up this subsystem benefit from storing and sharing the information contained in it.

- We extend an evaluation framework for open P2P systems proposed in the literature. We contribute a simulator that implements the evaluation framework and simulates the decentralized storage subsystem and the receipt-based accounting scheme, and is used to test mixtures of strategies that compete in evolutionary and non-evolutionary settings under various mobility models. The simulator can be extended with new strategies and new mobility models in a straightforward way.

- We use this simulator to show that our proposed strategy for encouraging cooperation in P2PWNC performs better than other strategies, and it leads to increased cooperation levels when evolutionary learning is possible.

- We design a simple vendor-neutral P2PWNC protocol that specifies the interactions of entities in a P2PWNC system. We also analyze the different digital signature schemes that can be adopted by our receipt-based accounting scheme.

- We implement the protocol and measure the performance of the implementation. The implementation runs on embedded, resource-constrained devices: the low-cost Linksys WRT54GS WLAN AP [30], and the QTEK 9100 WLAN-enabled cell phone [13]. We thus show that the protocol is implementable, and that P2PWNC can be used as a basis for a secure WLAN-based substitute to cellular telephony. (We prove the second part of the claim via a prototype VoIP application and a VPN and QoS architecture that is based on the Linksys APs themselves; however, these developments are beyond the scope of this dissertation; more information can be found in [16, 55] and on the P2PWNC project website [56].)

1.6 Dissertation outline

The remainder of this dissertation is organized as follows. Chapter 2 reviews the literature on incentive techniques, cooperation theory, and presents works related to the P2PWNC scheme. Chapter 3 defines 54 terms that are used throughout the dissertation. Chapter 4 presents the distinctive characteristics of the P2PWNC architecture and the main architectural entities. Chapter 5 presents the three algorithms that will be evaluated in Chapter 6; these are the reciprocity algorithm, the gossiping algorithm, and the bootstrap algorithm. Chapter 6 evaluates the three algorithms with simulation-based experiments; the chapter starts by introducing our evaluation framework and the P2PWNC benefit-cost model used in our simulator. Chapter 7 presents the P2PWNC protocol, with usage examples; it also presents the P2PWNC implementation (the first prototype) along with performance measurements. Chapter 8 discusses issues left open by previous chapters and outlines avenues for future work. Then, the conclusion of the dissertation follows.

Chapter 2

Literature Review

This chapter presents several proposals for incentive techniques, designed to encourage cooperation among selfish entities in peer-to-peer systems. Some of them are general economic frameworks that can be used irrespective of the specific type of resource that the peers share (storage space, CPU cycles, content, or bandwidth). Other incentive techniques apply only to specific types of resources.

The objective of this chapter is to introduce common architectural requirements, such as the need for central authorities and tamperproof modules. Chapter 4 refers back to these requirements when presenting the P2PWNC system model and its distinctive characteristics.

This chapter also presents the problem of *cheap pseudonyms* and the Sybil attack, as well as the problem of implicitly assuming cooperation in order to encourage cooperation. (For example, assuming that peers are willing to audit peers and punish those who free-ride, for the good of the community—instead of following the rational approach to free-ride on the efforts of other auditors.)

The chapter also introduces elements of Cooperation Theory [57] and summarizes two systems that have specific similarities to the P2PWNC proposal even though they do not focus on the problem of encouraging cooperation among selfish entities. Finally, the chapter summarizes related work on an analytic model for a P2P WLAN sharing scheme.

2.1 Sharing nicely

As an introduction to this chapter, we refer to an essay, appropriately titled "Sharing Nicely" [58], in which Yochai Benkler identifies WLANs in his list of "shareable goods." Shareable goods are goods that are (1) technically "lumpy" and (2) of "mid-grained" granularity. "Lumpy" means that they provision their functionality in discrete packages: a user cannot subscribe to a DSL connection that is less than some threshold capacity, or buy an IEEE 802.11b WLAN AP that provides less than 11 Mbps bandwidth at the physical layer. Therefore, once he provisions it, he has at a minimum a certain amount of bandwidth whether he needs all of it or not. "Mid-grained" means that there will be

relatively widespread private ownership of these goods, but the goods will systematically exhibit slack capacity relative to the demand of their owners.

The essay recognizes that technologies such as DSL/WLAN "[lower] the capital costs required for effective individual action [and allow] various provisioning problems to be structured in forms amenable to decentralized production based on social relations, rather than through markets or hierarchies." Although the essay takes the optimistic view that humans are in general willing to share, and that this type of *renewable goods* are ideal for sharing, it also mentions the possibility of excluding free-riders in order to provide the right incentives for sharing. The essay discusses the potential costs of selective exclusion, and that these costs may be a reason for owners to practice "open sharing."

In our system, we assume that the direct cost of selective exclusion is zero (because our algorithms handle it automatically); we assume that there is a non-zero cost involved with sharing, therefore, the owner can minimize costs by avoiding to serve free-riders who will not reciprocate (assuming their prior free-riding behavior is an indication of their future behavior). We present our specific benefit-cost model in Section 6.1.

2.2 Incentive techniques in peer-to-peer systems

In [59], a system that encourages sharing of storage resources is presented. The specific goal is to encourage peers to consume as much as they contribute. In storage terms, this means that a peer should only consume as much of the network's storage as it contributes for others on its local disk. The incentive technique used in this system relies on *auditing*. Every peer maintains a usage file, digitally signed by the peer, which is available for any other peer to read. The main components of this usage file are the *local* and the *remote* lists, which contain, respectively, the identifiers and sizes of all files that the peer stores locally on behalf of others, and the identifiers and sizes of the files stored remotely by this peer. Through an algorithm that checks the usage files on various peers, an *auditor* can detect peers that lie by advertising stored files they are not really storing. The disadvantage of this approach is that it requires auditors to expend effort for the good of the community without any direct benefit to them. The paper states, characteristically, "that since everybody, including auditors, benefits when cheaters are discovered and ejected from the p2p network, nodes do have an incentive to perform these random audits." However, this implicitly assumes that auditors are cooperative—the rational strategy for auditors would be to free-ride on the efforts of other auditors.

The KARMA system [60] is a general economic framework that encourages cooperation in self-organized peer-to-peer systems; it ties consumption to contribution through distributed accounting. More specifically, each peer is characterized by single scalar value, its *karma*, which is increased every time the peer contributes resources and decreased every time he consumes. KARMA is used to encourage cooperation in a file-sharing application. A peer's karma is tracked by a set of other peers, which are called the peer's *bank-set*. These peers allow or disallow a file exchange between two peers to proceed depending on the consumer's karma.

There are several potential problems with the KARMA approach. First, it relies on a Distributed Hash Table (DHT) structure that is used to assign peers to their bank-sets. Maintaining a DHT implicitly assumes cooperation among the peers in the DHT layer. Even if the DHT did function, the peers that are assigned to be a peer's bank-set will want to avoid this duty because there is nothing that they gain directly by holding the karma of other peers. Thus, the system may collapse when no peers are willing to take up bank-set duties. The third problem with KARMA is that when a peer first joins a KARMA network and discovers his bank-set, the bank-set allocates start-up funds to this peer (to bootstrap his participation in the economy). To join KARMA, peers need to solve a crypto-puzzle first, whose solution they include in their KARMA ID for all to see, proving that they spent time generating that ID. However, the crypto-puzzle that peers need to solve only limits the rate at which IDs can be created. By joining KARMA multiple times under multiple IDs, a peer can use them to falsely trade among them, and transfer the sum of the karma of these IDs to one ID, that is, earn karma without doing useful work for the KARMA community.

In [61], a system that encourages cooperation for packet forwarding among nodes in ad hoc networks is presented. The system assumes that every node is its own authority and need not cooperate with other nodes in the forwarding of packets for the good of the community. If all nodes do that, then cooperation would collapse and the ad hoc network would cease to function. To solve the problem, the paper encourages nodes to forward packets by using a currency called *nuglets*. This is not a currency in the form of digital tokens; rather, it is a counter that each node maintains, which increases whenever a node forwards packets for others and decreases whenever a node needs to send one of his own packets to the network. By requiring nodes to have positive nuglet counters in order to be able to send their own packets, cooperation can be encouraged. However, the (local) nuglet counter needs to be tamper-resistant otherwise selfish nodes can change its value arbitrarily.

To solve this problem the paper [61] relies on a tamper-resistant hardware module in each node, which maintains the nuglet counters of the nodes. These modules would be manufactured by a limited number of trusted manufacturers.

In [62], a scheme that is similar to P2PWNC is presented. It is an incentive scheme that encourages commercial operators of public WLANs to allow customers of other operators to access their networks. The scheme is exchange-based, in that the operators are paid "in kind": they earn the right for their own customers to access WLANs belonging to operators that participate in the scheme. As such, it is in principle similar to the peering agreements that are common among network operators. One difference is that the operators are given incentives to provide service of good quality because, at the end of each WLAN session, the client can report a satisfaction value to a *Trusted Central Authority* (TCA), which maintains *reputation* values for each operator that participates in the scheme. Therefore, the scheme is centralized; it also relies on an identity-certifying authority (again, the TCA) that distributes unique certified identities to the participating operators, so that recognition is possible and clients know which operator to rate. The cost of registering with a TCA may be too high for residential WLAN owners and, in our proposal, it is avoided because no authority oversees P2PWNC.

A work that is closely related to ours is on an exchange-based incentive mechanism for file sharing [63]. This scheme assumes an environment without any form of central authority; it relies on a multi-way exchange economy that is similar to the indirect debt paths that we will present in Section 5.1. Quoting from the paper [63]: "requests from peers that can provide a simultaneous, symmetric service in return (*exchange transfers*) are given higher priority. The service need not be directly to the provider (a *two-way exchange*), but more generally priority is given to peers who participate in *n-way exchanges* to which the provider currently belongs [which are] implemented as *rings* of *n* peers, where each peer is served by its predecessor and serves its successor in the ring." This scheme is thus a multi-way-exchange version of BitTorrent (see Section 1.4). Simulations and measurements on real network traces presented in the paper suggest that organizing such multi-way exchanges can in fact be practical in file-sharing networks.

Another proposal, which uses an exchange-based mechanism for sharing storage space, is *Samsara* [64]. Samsara does not rely on authorities and to encourage cooperation it requires that each peer that requests storage of another must agree to hold a *claim* in return—a placeholder that accounts for available space. After an exchange, each partner

checks the other to ensure faithfulness. Although the idea of constructing such symmetric exchanges is promising for a storage-sharing network, it is difficult to implement it in a system like P2PWNC because there is no equivalent way of holding claims if the resource to be exchanged is access bandwidth.

Reference [65] describes an incentive technique that is appropriate for routing in peer-to-peer structured overlay networks (that is, Distributed Hash Tables—DHTs). The *iterative* routing in DHTs that the paper presents works as follows. A node must forward a message to its destination, but is aware only of a specific number of neighbors in the overlay. They also have their own neighbors, and they know which of their neighbors can forward the message one step closer to its destination. If they are cooperative, they inform the requester. The paper examines a simple model where there is a constant cost involved with helping requesters. More specifically, a node "pays" 2 units every time it answers such a query. On the other hand, if it does not answer the query, the node does not pay any cost, but the requester will not be able to forward the message to its destination. If the requester continues from its neighbor, to neighbor's neighbor, and so on, and all neighbors cooperate (each one paying the cost of 2 units), then, if the message reaches its destination the original requester earns 40 units (of benefit). Reference [65] further assumes an all-powerful monitoring system that observes all such transactions, but with a probability that some transactions may go unnoticed. If a node does not cooperate in the manner we described above, it is marked "punished" by the monitoring system. Then, nodes may request information from the monitoring system, and may decide not to cooperate with punished nodes. Simulations show that this incentive technique can encourage nodes to cooperate most of the time, assuming a random matching game in which all nodes are both requesters and contributors of routing information. The simple benefit-cost function that is used has similarities with the one we will present for P2PWNC in Section 6.1.3.

Reference [66] presents the *EigenTrust* reputation system for peer-to-peer networks. The EigenTrust system can compute a global trust value for a peer in the network by calculating the left principal eigenvector of a matrix of normalized local trust values. The objective of EigenTrust is to provide incentives to peers to share files that are authentic. Peers who share inauthentic files (with respect to their advertised name) are reported to the reputation system when their "victim" rates their transaction as negative. On the other hand, peers who share authentic files receive positive ratings from their requestors. Although these values are subjective, EigenTrust computes a global reputation value for each peer by

weighing the local trust values assigned to this peer by other peers, weighted by the global reputations of these peers. The algorithm can work in both centralized and distributed versions, where instead of having one central server doing the calculations of trust values, this task is distributed across all EigenTrust peers. The global trust value of a specific peer is calculated by his set of *score managers*. The score managers of a peer are assigned to him using a DHT, which maps the ID of the peer to a certain number of other peers. The potential weakness here is that DHTs assume *cooperative* nodes at the overlay layer, and it is not clear how this assumption aligns with the model of *malicious* nodes who share inauthentic content in order to minimize their costs (see also the analysis in Section 4.3.6).

Similar to EigenTrust is the NICE system [67] (to which we will return in Section 4.3.6). In NICE, peers store *cookies* that were given to them by peers who successfully cooperated with them in the past. NICE peers store these cookies and show them to other peers to prove that they are good cooperators.

2.3 Tit-For-Tat

Cooperation theory [57, 68, 69] studies the conditions under which cooperation can emerge in systems with self-interested participants when there is no centralized control. The *repeated prisoner's dilemma* [57] game provides a useful model. In the prisoner's dilemma game, two players meet and they each have a choice of moves: "cooperate" or "defect'. Assuming the simplest form for the game's payoff matrix, we have that if both players play "cooperate," a banker pays both of them one dollar (R = 1, the reward for mutual cooperation). If one plays "cooperate" while the other plays "defect" the defector receives two dollars from the banker, while the cooperator has to *pay* one dollar (T = 2, the temptation to defect, and S = –1, the "sucker's" payoff). If both play "defect" then they both get nothing and they pay nothing (P = 0, the punishment for mutual defection). Actually, the numbers themselves are unimportant. The game is still the same as long as $T > R > P > S$, and $R > (S + T) / 2$. It is interesting to note that in the one-shot prisoner's dilemma game, defection is the only move that a selfish and rational participant could play (assuming that prior communication between the players is impossible). No matter what the opponent plays, defection is always optimal.

In [57], simulations were conducted with various computer programs, each following a different algorithm, which engaged in the repeated prisoner's dilemma among themselves (matched in a round robin fashion, where each program would play one match

with every one of the others, for several rounds). Some of the submitted programs were quite complex. Some obvious strategies for the prisoner's dilemma include ALLC and ALLD [57]. The first specifies unconditional cooperation (all the moves are "C" for "cooperate," hence its name). The second is, ALLD, specifies unconditional defection (all the moves are "D" for defect).

The strategy that emerged as the winner from these games was Tit-For-Tat (TFT). TFT is a simple strategy that specifies cooperation on the first move with a new opponent, and then, if the opponents meet again, it specifies that the follower of TFT must play whatever his opponent played when they last met. Therefore, a player cooperates as long as the other player cooperates, and defects, just once, in retaliation for a defection. Thus, TFT is *forgiving* [57]. TFT is also *nice*, in that it starts by cooperating. We will revisit the TFT strategy as a baseline case in what follows.

2.4 The Sybil attack

In [70], the problem known as the *Sybil attack* was formally identified. The Sybil attack in peer-to-peer networks is an important security threat, as well as a threat to cooperation. The main argument of [70] is that it is practically impossible in a distributed computing environment for unknown remote entities to present convincingly distinct identities. If there is no logically centralized trusted authority that can vouch for the one-to-one correspondence between entity and identity (for example, by signing a certificate that binds the real-world identity of the node with his electronic identity), then a node can create many distinct electronic identities. In peer-to-peer systems that attempt to encourage cooperation, this allows an entity to engage in *false trading* with itself. The simplest form of false trading is for the *Sybil IDs* of a node to advertise that the real ID of a node is a good contributor to the system. This is problematic, for example in online auction environments, where peers rely on the testimonials of other peers in order to judge and pick their prospective transacting partners. Another approach to the Sybil attack involves communities that rely on a micro-payment scheme and supply start-up funds to new entrants in order to bootstrap cooperation in the system (see the KARMA approach in Section 2.2). If a peer can join the same community using different IDs, he can transfer all the funds from the many Sybil IDs to the one "real" ID by engaging in false trading (we assume that in a decentralized system, transactions between two peers cannot be monitored by third parties). This allows free-riders to earn an unbounded amount of electronic currency, which they can

use in the system to consume without contributing resources (in order to gain currency), thereby defeating the purpose of the currency scheme.

Reference [71] proposes a set of incentive techniques for encouraging cooperation in peer-to-peer communities. The paper acknowledges the possibility of Sybil attacks. In the model of the paper, the peers contribute and consume services from one another. Every time a contribution takes place, it costs to the contributor one unit, and it benefits the consumer seven units. Although arbitrary, these numbers serve to underline a basic point: this is the assumption that social welfare (see Chapter 3 for a definition) is increased after every transaction in a peer-to-peer community. The results presented in the paper would be qualitatively the same for other benefit-to-cost ratios.

The peers engage in rounds of transactions. The objective of the paper is to find an appropriate *reciprocative decision function*, which can be adopted by the peers to guide them in their cooperation decisions. The paper shows through simulations that their specific technique can drive a system of selfish and rational users to nearly optimal levels of cooperation.

The specifics of their model follow. Every peer keeps record about all the interactions that occur in the system, regardless of whether he was directly involved in them or not. This allows the peers to take advantage of the experiences of others. This only requires that someone has interacted with a particular peer for the entire community to observe it. The problems with this scheme, which the paper calls *shared history*, are two: First, in order for all the peers to be able to view the history, some type of logically centralized repository of information is required; this can be a central server (which is undesirable), or a Distributed Hash Table (DHT—see Section 1.4.3), built using the peers themselves. Second, assuming Sybil attacks are possible, the shared history can be polluted by free-riders. Free-riders can create multiple identities, choose one of them, and have the other identities falsely trade with this identity in order to make it appear cooperative.

The paper addresses the second problem by introducing the concept of *maxflow-based* reputation. Maxflow-based reputation uses the maximum flow graph-theoretic algorithm. In maxflow, given a directed graph with weighted edges, we can find what is the greatest rate at which "material" can be "shipped" from the source to the target without violating any capacity constraints. The paper applies the maxflow algorithm to a graph whose vertices are peers and the edges represent the services that peers have received from each other. The paper illustrates the usefulness of the maxflow algorithm in a scenario

where peers can obtain multiple IDs, who (falsely) report having received service from each other. When Peer P decides to see if a Peer C (who used multiple IDs in a Sybil attack) is a good cooperator, the false trading of Peer C with his Sybil IDs will not affect the result of the maxflow computation from P to C, which will still be zero. This is illustrated below. (This figure is based on the corresponding figure in the paper.)

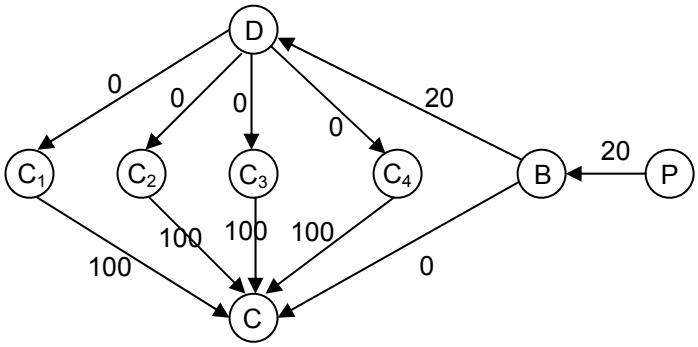

Figure 2.1 Using the maximum flow algorithm to detect free-riders who perform the Sybil attack. The maximum flow from P to C is zero, even though C created Sybil IDs that faked evidence of contribution (edges with weight 100). This is because no Sybil ID, nor Peer C actually contributed service to any other peer.

In Figure 2.1 above, C_1, C_2, C_3, C_4 represent the Sybil IDs that Peer C created. Peer P is a regular peer, and so are Peers B and D. The Sybil IDs declare that they have received service from Peer C—this is modeled in the graph with edges of weight 100 that point *to* C. However, the maxflow from P to C is still zero, because none of the Sybil IDs is real, and therefore no Sybil ID has contributed to other peers (such as for example, Peers B and D). C himself is a free-rider and has never contributed to Peer B (hence the edge with weight zero from Peer B to Peer C).

In the paper, they proposed that the benefit a Peer P has received (indirectly) from a Peer C (or, conversely, the debt that Peer P "owes to" Peer C, is the maxflow from P to C). Then, they propose and evaluate the effectiveness of the following decision algorithm: When Peer C requests service from Peer P, Peer P should contribute service with probability:

$$p = \min\left(\frac{mf(P \to C)}{mf(C \to P)},\ 1\right)$$

This probability *balances* the two debts (the one P owes to C, and the one C owes to P). The decision algorithm that uses this probability is shown through simulations that lead the community to near-optimal levels of cooperation even with peers that perform the Sybil attack and falsify evidence of contribution. Moreover, this algorithm encourages peers to contribute as much as they consume. The more they depart from this 1-to-1 ratio, the more the probability p will be less than one, and the less other peers will cooperate with them.

The evaluation framework of this paper is the basis of the evaluation framework we will present in Section 6.1. Moreover, the reciprocity algorithm that we present in Section 5.1 builds on this decision function, exposing one specific vulnerability, and amending it. And the gossiping algorithm that we present in Section 5.2 is a solution to the shared history problem that does not require central servers or DHTs to function. (We discuss why DHTs are undesirable in Section 4.3.)

In [72], the authors of [71] continue the analysis of how the Sybil attack can be damaging to cooperation levels. The paper proves formally that the Tit-For-Tat (TFT) strategy (see above, Section 2.3) is ineffective in an environment where peers can change IDs freely. Although the formal proof is involved, the intuition is simple: because TFT specifies "cooperate" every time it interacts with a peer it has not seen before, then an obvious way to attack TFT is to constantly change IDs before interacting with TFT followers. This way, free-riders will receive the benefit of cooperation, and they could continue defecting indefinitely (see also our analysis in Section 5.1.5).

Finally, the effect of the Sybil attack (before it was formally identified as such) on cooperation levels was analytically derived in [73]. Reference [73] proved that "suspicion of strangers is costly to society." The paper proves that strategies that cooperate with strangers are not stable, and that no strategy can do substantially better than punishing all newcomers. They note though that a large degree of cooperation can emerge, through a convention in which newcomers "pay their dues" by accepting poor treatment from players who are already established in the system. We take this for granted in our P2PWNC scheme, and specifically design our bootstrap algorithm (see Section 5.3) with the "pay their dues first" philosophy in mind.

2.5 Other similar schemes

Reference [74] proposes a self-organized public-key management scheme. The scheme does not focus on incentives issues; rather, it represents a fully decentralized

Certification Authority (CA), suitable for mobile ad hoc networks. In mobile ad hoc networks, connectivity is unpredictable and access to CAs could be difficult. CAs, however, are useful in *Public Key Infrastructures* (PKIs): they publicize the (certified) public keys of users, keys which are often difficult to find but which are required if one wants to send to a user an encrypted message or to verify a user's signature on a document. A certified public key is a public key that is bound to a specific identity—such as an email address—by the digital signature of the issuer of the certificate (usually a CA, whose own public key is assumed to be well-known, and therefore the CA's signature is always verifiable). Another important function of (online) CAs is that they maintain online *Certificate Revocation Lists* (CRLs) with the certificates that the CA has revoked prior to their normal expiration date.

In the system in [74], no centralized CA is required. The decentralized PKI proposed is similar to PGP [75] and P2PWNC in that users/nodes generate their own public-private key pairs; also, the users, and not a central CA, sign public key certificates for other users, in a manner similar to PGP. These certificates form a certificate graph, similar to the P2PWNC receipt graph. Vertices in this graph represent identity-public key bindings and edges represent certificates, that is, documents signed using the private key of the source vertex, which indicate that the source vertex believes that the identity-public key binding represented by the target vertex is correct. Usually such a certificate will be generated when a person meets another person face to face; one person then presents his public key, and the other, the certifier, generates and signs a public key certificate for that key, being certain that the public key-identity binding he signs is correct because of the secure face-to-face communication. Certificate paths on the graph, starting from an identity-public key binding, indicate that the source vertex ultimately believes that the target identity-public key binding is correct, as long as all the intermediate certificates are verifiable using the previous public key in the chain. This holds only if the assumption of *transitive trust* holds, that is, the user trusts the targets of edges whose sources he already trusts. This way, if a user discovers a certificate chain connecting him (as the source) to any other vertex, he can be sure that the identity-public key binding represented by that vertex is correct. The problem then becomes how to find a certificate chain without a centralized authority to store all the certificates—similar to the problem of a contributor in P2PWNC who tries to find one or more debt paths from a source vertex (representing himself) to the target vertex (representing the public key of the consumer who is requesting service).

Reference [74] presents an algorithm called Maximum Degree, which allows users to discover such certificate chains. The paper shows through simulations that two users, even when unknown to each other, can verify with high probability each other's public keys when they meet by *merging* their local certificate repositories and searching for certificate paths in the merged repositories. This algorithm specifies a certificate replacement policy in the repositories and it was the inspiration for the P2PWNC gossiping algorithm. One semantic difference is that, with certificates, a single chain connecting two vertices is enough. In the P2PWNC receipt graph, more chains mean more (indirect and direct) debt, and the amount of debt is important in P2PWNC because it is used to compute the amount of cooperation ultimately offered.

Another difference is that in the design of [74] a node is assumed to know all the certificates that were issued by others for his public key (this is equivalent to edges that point *to* him) and to know also all the certificates that were issued by him (equivalent to edges that start *from* him). In P2PWNC, we assume that peers know the receipts pointing to them (having earned them directly through the contribution of WLAN service—see the receipt generation protocol, Section 4.6.2), but P2PWNC peers are not aware of their consumption actions (equivalent to edges that start *from* them); these are performed by one or more *members* of a P2PWNC team/peer.

Reference [76] proposes another system with similarities to P2PWNC called *Self-Organizing Wireless messaging nEtwoRk* (SOWER). The system's objective is to provide a free alternative to the cellular-based Short Message Service (SMS). In a user-owned SOWER network, the users/owners must first install special "home" devices—essentially, IEEE 802.11 APs—that they then share with other (mobile) users/owners of the system who are within range, with the understanding that they can do the same when the roles are reversed. These home devices (1) form a citywide wireless mesh network in the ISM band, and (2) act as gateways for mobile devices that wish to send messages to other mobile devices. The messages traverse the SOWER network hop-by-hop to reach the home device of the target user—thus making finding users easier: if the home device tracks the current location of its (mobile) owner via the same mesh network, the message can be forwarded to its final destination directly (this is similar in principle to how users are located in Mobile IP-based communication via the Home Agent); otherwise, the message can be stored for later delivery.

Unlike P2PWNC, the SOWER algorithms do not include incentive techniques to stimulate cooperation. Because the home devices are energy-rich (powered by the fixed

network), and because no obvious security threats to an owner's home computer network exist, the paper assumes that their owners will simply keep these devices turned on. The paper, however, does make one incentives-related observation: by leaving the devices turned on, their owners, when mobile, can also receive the incoming messages that are addressed to them through the "home agent" functionality that we described above. This is, however, equivalent to assuming that the devices are tamperproof; otherwise, a user could reprogram them, keeping the home agent functionality but turning off the AP functionality. The paper hints at this attack and suggests that a solution relying on "virtual money" may be required; such a solution is the focus of our work, in a similar context (citywide user-owned networks), but for a different application (wireless Internet access).

In [77], an important result concerning P2P WLAN sharing schemes is derived. Reference [77] considers the problem of provisioning a public good. The public good is provisioned with private contributions, which come at a cost to the contributors. The same contributors, though, derive benefit from the public good, once it has been provisioned. This paper focuses on two applications, the second of which is the provisioning of *coverage* by peers who operate WLANs. If we assume that congestion is not issue, WLAN coverage is indeed a public (that is, non-rivalrous) good, in that the amount of coverage that is enjoyed by a participant does not reduce the amount of coverage available to other participants.

The paper shows that, if exclusions from participating in the scheme are possible, then, as the number of participants becomes large, a very simple provisioning scheme that requires that each participant provides a *fixed* contribution (of coverage, in the WLAN application) can lead the system to nearly optimal efficiency.

This is an important result because it shows that simple provisioning policies can be effective, and one need not engage in complex calculations in order to derive, for example, *personalized* contributions for each participant.

Chapter 3

Definitions

This chapter defines 54 terms grouped in 36 entries. We use these terms throughout this dissertation. The terms belong to the scientific disciplines of Economics and Computer Science (Security and Peer-to-Peer Networking), and some are considered well known. Nevertheless, these terms have been overloaded with multiple meanings because of their frequent use in various contexts. Because they are central to our thesis, a precise, concise, and unambiguous (re-)definition is required. Note that even if certain definitions seem counterintuitive, the use of these terms in the context of this dissertation is consistent with the definitions that appear in this chapter.

We accompany the definitions with examples as well as pointers to the relevant sections of this dissertation, where appropriate. If, in a definition, a term appears italicized, then the definition for the italicized term is also included in this chapter.

A reader could continue directly to Chapter 4; however, the style of this chapter is introductory and the definitions delineate the central concepts that will follow.

Accounting The process of recording (a summary of) the details of service *consumption* that usually follows successful *authentication* and *authorization*; for example, the process of recording the amount of bytes that a WLAN client transfers over a wireless Internet connection. (See also P2PWNC receipt generation protocol in Section 4.6.2.)

Altruism The opposite of *selfishness*; the *irrational* practice of *cooperating* (usually at a *cost* to oneself) with anyone asking for help. Also known as *unconditional cooperation*. We regard altruism as *irrational* in the sense that altruists do not attempt to maximize their *net benefit*, although we could have considered more complex models in which altruistic actions are beneficial in themselves (the "warm-glow" effect—the pleasure of giving). Altruists inadvertently also *cooperate* with *free-riders*, and the presence of altruistic *strategies* in *evolutionary games* usually allow *free-riding strategies* to persist, leading to reduced *social welfare*. (See also P2PWNC evolutionary games in Chapter 6—Evaluation.)

Authentication, Successful Authentication, Mutual Authentication Authentication is the process through which an entity with *identity* "A" attempts to convince another entity *B* that it is indeed entity *A* and not an imposter claiming to be *A*. A usual authentication method, the password method, requires *A* to supply to *B* *identity* "A" accompanied by a correct password, that is, a password associated with the particular *identity* in the authentication database at *A*'s administrative domain. Another authentication method is for an entity with *public key* "A" (which we can assume is equivalent to *identity* "A" above) to generate and to supply to *B* a digitally signed document that can be verified using *public key* "A". Successful signature verification means that the document was signed using the *private key* that corresponds to *public key* "A"; and only the entity with *public key* "A" possesses the corresponding *private key* (this is a standard cryptographic assumption). Both methods' assumptions are invalidated if entities other than *A* gain possession of the password or *private key* respectively, or if a *replay attack* is conducted. Note that the password method relies on *B* also knowing the password. Note also that the process of supplying the password in the password method is not necessarily susceptible to *eavesdroppers* or *replay attacks*: The communication channel connecting the two entities can be secure, or a challenge/response protocol (to avoid replay attacks) can be used over an insecure channel.

Successful authentication is the authentication process that results in entity *A* convincing *B* that *A* is indeed the entity with *identity* "A."

Mutual authentication is the process through which the two entities attempt to convince each other of their respective *identities*.

Authority A logical center, which could be implemented in a physically *centralized* or *decentralized* manner, with powers over a specific *community*. These powers can include one or more of the following: (1) the power to control which members join the *community*; (2) the power to reward or to punish *community* members; (3) the power to be recognized as the sole repository of definitive information for the *community*, such as the *community*'s *history*.

Authorization Usually follows successful *authentication* and is the process of granting an entity with access to specific services. For example, allowing an entity to access an Internet connection. To lower the probability of unauthorized access to resources via *session*

hijacking, *authentication* must be repeated periodically. (See also P2PWNC receipt generation protocol in Section 4.6.2.)

Benefit, Cost, Net Benefit, Social Welfare The benefit of an action is a non-negative number that conveys the amount of satisfaction received by performing the action. The cost of an action is a non-negative number that conveys the amount of dissatisfaction received by performing the action. The net benefit of an action (or of a series of actions) is the benefit (if any) of the action (or series of actions) minus the cost (if any) of the action (or series of actions), and can be positive, zero, or negative. These abstractions are not always meaningful but they can be useful in comparing the effectiveness of various *strategies* over time (that is, over a series of actions). This is because we can add up benefits, costs, and net benefits over time (that is, over a series of actions). Social welfare is the sum of the net benefits of a *community*'s members.

Centralized, Decentralized Terms used to describe a system's architecture and implementation. A (logically) centralized architecture can have a (physically) centralized or (physically) decentralized implementation. A (logically) decentralized architecture can only have a (physically) decentralized implementation. A (logically) centralized architecture relies on an *authority* by definition. In a (fully) decentralized architecture no *authority* exists. (See also P2PWNC architecture in Chapter 4.) Sometimes, decentralized architectures are referred to as "fully decentralized" in the literature in order to distinguish them from decentralized architectures that rely on an identity-certification *authority* that is used only to distribute certified *identities* to newcomer nodes. For our purposes, a decentralized architecture is equivalent to a fully decentralized architecture.

Collusion, False Trading, Naïve Collusion, Moles, Front Peers Collusion is the *cooperation* of two or more *rational* entities that attempt to achieve positive individual *net benefit*. Collusion implies a side channel, outside the confines of the normal communication channels of the *community* in question, a side channel that allows the colluding entities to communicate and synchronize their actions.

In practice, we assume that colluding entities mainly engage in false trading. That is, they create fake evidence of *contribution* and *consumption* showing that some of the colluding entities *contributed* service to other colluding entities in an attempt to pollute the

system's *history* and convince followers of *strategies* that rely on this *history* that some of the colluding entities where, in fact, actual *contributors* of service to the *community*.

In naïve collusion, the colluding entities never *contribute* service to the *community*, but rely only on false trading. Therefore, in naïve collusion, all colluders are effectively *free-riders*. In more sophisticated collusion scenarios, the colluders can include one or more moles or front peers; these are colluders that *contribute* service to the *community* but also collude with other colluders.

Community A set of entities that use a specific peer-to-peer application in order to *contribute* and *consume* a *resource*. A successful community is a community that generates positive *social welfare* over time. A successful community can, however, contain at any one time a mixture of dissatisfied entities (in the sense that the *net benefit* of these entities over time is negative) and satisfied entities (in the sense that the *net benefit* of these entities over time is positive).

Conditional Cooperation See *Cooperation*, *Unconditional Cooperation*, *Conditional Cooperation*.

Confederation Another term for *community*. The term confederation is used to emphasize a *decentralized* architecture (as opposed to a "federation," where a logical center, or *authority* exists, and has more powers than the corresponding *authority* in a confederation).

Consumer, Contributor "Consumer" is a transient role that a member of a *community* adopts; it is during this time that a consumer engages in a *consumption* action, at a *benefit* to oneself and at a *cost* to some contributor. "Contributor" is a transient role that a member of a *community* adopts; it is during this time that a contributor engages in a *contribution* action, at a *cost* to oneself and at a *benefit* to some consumer. (See also Chapter 5—System Algorithms, and the evolutionary games of Chapter 6—Evaluation.)

Consumption, Contribution A consumption action is the action of a *community* member who enjoys a *community*'s *resource* at a *benefit* to oneself and at a *cost* to some *contributor*. A contribution action is the action of a *community* member who contributes the *community*'s *resource* to some *consumer*, at a *cost* to oneself and at a *benefit* to the *consumer*.

Cooperation, Unconditional Cooperation, Conditional Cooperation Cooperation is an individual behavior of a *community* member that incurs personal *cost* in order to engage in a joint activity that confers a (variable) *benefit* (usually exceeding the *cost*) to some other member of the *community*. In the context of our P2P WLAN sharing scheme, two entities cooperate when the prospective *contributor* decides to *contribute*, at a *cost* to himself, and the prospective *consumer* (who was the one requesting service) *consumes*, at a *benefit* to himself.

Unconditional cooperation is equivalent to *altruism* and is the principle followed by prospective *contributors* who cooperate with any prospective *consumer* that requests it.

Conditional cooperation is the opposite of unconditional cooperation. Conditionally cooperating *strategies* may condition their cooperation decisions on random coin tosses, on the *community*'s *history*, on the *identity* of the prospective *consumer*, or on combinations of the above and other factors.

Cost See *Benefit*, *Cost*, *Net Benefit*, *Social Welfare*.

Decentralized See *Centralized*, *Decentralized*.

Denial-of-Service (DoS) Attack The *irrational* practice of interfering (at zero *benefit* and at positive *cost* to the interferer) with service provision in order to deny others part (or all) of the *benefit* of *consuming* the *resource* that the specific service provides. Common DoS attacks include: (1) jamming a wireless network (physical-layer DoS attack); (2) repeatedly trying to initiate a link-layer connection (for example, generating an "association-request flood" in the case of IEEE 802.11) with a wireless hub (link-layer DoS attack); (3) flooding an Internet router or Internet end-host with IP packets (network-layer DoS attack); (4) repeatedly trying to initiate a transport-layer connection with a remote server (for example, generating a "TCP SYN flood" in the case of TCP—transport-layer DoS attack); (5) repeatedly sending application-layer requests that cause congestion at an application server's CPU, storage, or access bandwidth (application-layer DoS attack).

Double Spending In electronic micro-payment schemes, the practice of using the same, already spent, digital token as payment in more than one *transaction*. In non-electronic scenarios this is equivalent to forgery; the difference from electronically mediated

transactions that use tokens is that tokens can easily be copied. Even if a central bank has issued these tokens and signed them with its *private key*, and even if each token has a unique serial number, double spending is still possible if *consumers* use the token to pay in *transactions* involving various *contributors*. If the *contributors* do not synchronize token information among themselves, it may be difficult to know if a digital token has been "doubly spent" somewhere else. That is why, fundamentally, all electronic micro-payment schemes that adopt the token paradigm require an online *authority* to check for double spending; the *authority* must track the current location of tokens in the system (that is, which participant has which token). The advantage of using tokens instead of an online bank that simply maintains account balances is that, with tokens, *transactions* can proceed even when the bank is offline, but at a risk that some double spending may occur.

Eavesdropping Attempting to compromise a session's communication *privacy*. Eavesdroppers do not alter the content of a session, nor do they inject new content in the session. Eavesdropping can be defeated if the session in question is encrypted, for example using end-to-end encryption where the two communicating parties share a symmetric encryption key that the eavesdropper does not possess.

Exchange (direct/two-way exchange, indirect/multi-way exchange, inter-temporal exchange) In the context of this dissertation, a direct (or two-way) exchange occurs between two members *A* and *B* of a *community* when *A contributes* to *B* (at a *cost* to *A* and at a *benefit* to *B*) and, later, *B contributes* to *A* (at a *cost* to *B* and at a *benefit* to *A*). If we assume a constant universal *benefit b* for *consumers* in *transactions*, and a constant universal *cost c* for *contributors* in *transactions*, then, if $b > c$ the *social welfare* of the two-member group is positive at the end of the two *transactions* and, more importantly from a *selfish* point of view, so is individual *net benefit*.

In indirect exchanges (or multi-way exchanges), *B* need not *contribute* directly to *A*, but may *contribute* to some other member *C*, which may return the favor later by himself *contributing* to A, thereby completing a three-way exchange ring and forming a three-member group in which *social welfare* is positive, and in which all group members are satisfied (that is, individual *net benefits* are also positive). In the context of this dissertation, all these exchanges are inter-temporal exchanges in the sense that the *transactions* that comprise them do not occur at the same moment in time.

Evolutionary Games In evolutionary games, members of a *community* that follow various *strategies* engage in a series of *transactions* as prospective *consumers* and prospective *contributors*. In the context of this dissertation, the outcome of a *transaction* depends on the *strategy* followed by the prospective *contributor* and on the *community*'s *history*, but not on the *strategy* followed by the prospective *consumer*. The outcome can range from complete non-*cooperation* (if, for example, the prospective *contributor* follows a *free-riding strategy*), to full *cooperation*, or to any amount of *cooperation* in between (see Section 6.1 for more specific definitions relating to our evolutionary games). In the context of this dissertation, different amounts of *cooperation* may result in a different *benefit* for the *consumer* but they always incur the same *cost* to the *contributor* (see also Section 6.1).

In evolutionary games, the dual forces of *learning* and *mutation* cause members of a *community* to evolve, that is, to switch to using other *strategies*. The *social welfare* of the *community* depends on the mixture of *strategies* at any one time and is an important *community* metric: we can observe how the *social welfare* of a *community* changes over a series of member actions (that is, over time).

False Trading See *Collusion*, *False Trading*, *Naïve Collusion*, *Moles*, *Front Peers*.

Free-Riders, Second-Order Free-Riders Free-riders are *rational* members of a *community* who *consume* but never *contribute* in an attempt to maximize their *net benefit*. Second-order free-riders are members of a *community* who *cooperate* with free-riders, thereby encouraging free-riders to persist. *Unconditional cooperators* are second-order free-riders because they free-ride on the efforts of *conditional cooperators*, who do not cooperate with free-riders (and give free-riders an incentive to *contribute*). *Conditional cooperation* may require effort on the part of *conditional cooperators*—for example, the effort that is required to identify a free-rider for what he is. Second-order free-riders attempt to minimize the *cost* that punishing free-riders entails. If the only possible punishment is refusal to *cooperate*, there may still be *cost* associated with correctly identifying free-riders because this may require an examination of the *community*'s *history*.

Front Peers See *Collusion*, *False Trading*, *Naïve Collusion*, *Moles*, *Front Peers*.

History (short-term, long-term) In the context of this dissertation, the history of a *community* is the set of all *transactions* that have been *accounted* for. History may contain records of *transactions* that are the result of *false trading*.

History can be short-term or long-term, depending on how far back in time the *transactions* in the history refer to. Long-term history requires more memory to store than short-term history. In addition, with short-term history, earlier *contribution* actions of *community* members are forgotten, which encourages continuous *contribution* on their part assuming they wish to be recognized as good *contributors* at any random instant in time in which community history is examined.

Incentive Technique An incentive technique is any aspect of a system's operation that directly addresses user *selfishness* and *rationality* by giving the users the right incentives to complete an action they would otherwise consider *costly* and, being *rational*, would try to avoid. Incentive techniques usually assume that the software and hardware modules that implement the functionality of the system cannot be trusted to follow the designer's specifications because *selfish* peers may find ways to alter this functionality if it is in their interest. Here, we always assume that the (benevolent) designer has the goal of maximizing *social welfare* in mind.

Identity Any tag that can be represented in digital form and can be used to name an entity. An entity can have many identities. Free identities (also known as cheap pseudonyms) are identities that can be generated at low or zero *cost* to the entity (for example, they are not issued or need to be certified by an *authority* and can be generated by the entity itself).

Irrationality See *Rationality*, *Irrationality.*

Learning, Mutation In *evolutionary games* the process through which members of a *community* switch to using other *strategies.* This happens according to two probabilities, the "probability to learn," and the "probability to mutate" (see also Section 6.1.6). With mutation, a new *strategy* is chosen uniformly at random from a universe of possible *strategies*. With learning, the new *strategy* will be chosen depending on the *strategy*'s rating, which is a function of the *net benefit* of the *strategy*'s followers over time. High-rated *strategies* are chosen more often than low-rated *strategies*. This assumes that a *strategy*'s

rating can be observed "out-of-band" and that the "probability to learn" models the communication that occurs in the real world among *community* members who exchange experiences on how specific *strategies* influenced their *net benefit* over time.

Malicious An *irrational* attacker. The term is usually associated with DoS attacks that bring no *benefit* to the malicious attacker, only *cost*. At the same time, DoS attacks do not allow others to enjoy the *resource* of the *community*, which may lower their *benefit* increase to a very small number, or even to zero.

Moles See *Collusion*, *False Trading*, *Naïve Collusion*, *Moles*, *Front Peers*.

Naïve Collusion See *Collusion*, *False Trading*, *Naïve Collusion*, *Moles*, *Front Peers*.

Net Benefit See *Benefit*, *Cost*, *Net Benefit*, *Social Welfare*.

Privacy (identity privacy—anonymity, location privacy—untraceability, communication privacy—confidentiality) *Identity* privacy means hiding your chosen or your real-world *identity* from "others" while engaging in *transactions*. Location privacy means hiding your location from "others" while engaging in *transactions*. Communication privacy means hiding the content of your communication sessions from *eavesdroppers*. In the first two types of privacy, the exact meaning of "others" depends on the application; thus, "privacy" is a heavily overloaded term.

Public Key, Private Key In the various public key cryptography schemes (RSA, ECC, and others—see Chapter 7) each entity generates a public-private key pair that is guaranteed with high probability to be unique. The private key is then kept secret and is used to sign digital documents (that is, to sign a hash of the digital document in question; signing is the act of encrypting the constant-length output of the hash function with a private key and appending the result to the original digital document). Any other entity in possession of the entity's public key can verify that the document was indeed signed by the entity who is in possession of the corresponding private key, and that the document has not been altered (all these, assuming the hash function used is cryptographically secure also).

Public Good, Club Good A public good is a non-rivalrous, non-excludable *resource*, such as national defense and clean air (these are common examples). The *resource* of our P2P WLAN sharing scheme (which is wireless Internet bandwidth), however, is a club good, a non-rivalrous (because we will assume no congestion) but excludable public good. The *resource* is excludable because an *authentication* procedure can *authorize consumption*, for example, only if the system's *history* identifies the prospective *consumer* to be also a *contributor*.

Rationality (forward-looking, shortsighted), Irrationality Rationality is the distinctive characteristic of a *community* member who tries to maximize *net benefit*. Irrationality is the distinctive characteristic of a *community* member who does not try to maximize *net benefit*.

Forward-looking rationality specifies maximizing total net benefit over extended periods. Shortsighted rationality specifies maximizing total net benefit over shorter periods.

Replay Attack A common security attack that consists of capturing and replaying messages (equivalently: frames, packets, segments) in an attempt to bypass normal *authentication* procedures.

Resource The specific service (for example, high-speed Internet access over WLANs and DSL/Cable connections) that a *community* provides to its members. Members acting as *consumers consume* the service (at a *benefit* to themselves) and members acting as *contributors contribute* the service (at a *cost* to themselves).

Self-Organization Full self-organization is the same as full *decentralization*; that is, a logically and physically *decentralized* system architecture, where no *authorities* exist, not even to assist nodes/peers who first join the system/*community* (for example, by certifying their *identities*). In our context, a fully self-organizing *community* must also be an open *community* because it has no way to control who is eligible to join it and who is not.

Selfishness (or self-interest) The *rational* practice of *community* members who avoid helping others in an attempt to minimize their *costs*.

Session Hijacking A session is said to have been hijacked when, following successful *authentication* and *authorization*, another entity intervenes and *consumes* the *resource* represented by the ongoing *transaction* without this entity being the entity that went through the original *authentication* and *authorization* process.

Social Welfare See *Benefit*, *Cost*, *Net Benefit*, *Social Welfare*.

Strategy An algorithm followed by *community* members. The algorithm may or may not rely on *community history*, and it guides members' *cooperation* decisions.

Sybil Attack An entity is conducting a Sybil Attack if it obtains multiple *identities* in an attempt to *falsely trade* with itself.

Transaction A *contribution* action from the point of view of the *contributor* and a *consumption* action from the point of view of the *consumer*, which is *accounted* for, and its record is then stored in the *history* of the *community*. Whenever a transaction goes through, we say that the *contributor cooperated* with the *consumer*.

Unconditional Cooperation See *Cooperation*, *Unconditional Cooperation, Conditional Cooperation*.

Chapter 4

System Model and Architecture

This chapter presents the system model that underlies the proposed P2P WLAN sharing scheme. Along with the next chapter (Chapter 5—System Algorithms), they represent the complete design of the system called *Peer-to-Peer Wireless Network Confederation* (P2PWNC).

4.1 The P2PWNC system as a peer-to-peer system

The P2PWNC system encompasses all entities, protocols and algorithms that realize a P2P WLAN sharing scheme using readily available technology. P2PWNC is a Peer-to-Peer (P2P) system. Chapter 1 discussed P2P *file-sharing* and *CPU-sharing* systems. Below is a definition of P2P systems that is generic enough to encompass most P2P systems (including P2PWNC), and specific enough to highlight those aspects of P2P systems that make them different from client-server systems. This section also serves as an introduction to Section 4.3, which discusses the distinctive characteristics of P2PWNC compared to other P2P systems.

P2P systems, including P2PWNC, are electronically mediated communities of users who, with the help of appropriate software and hardware modules, cooperate for mutual benefit without centralized control. A basic principle of P2P systems is to preserve user autonomy while maintaining system reliability. P2P systems make use of otherwise underutilized low-level resources such as access bandwidth, processing capacity, storage space, and files, all of which are *shared* among peers. Aspects of this definition are elaborated below.

Shared resources All P2P systems involve the sharing of resources. Examples include low-level resources such as storage space, processing capability, and access bandwidth. Resources also include higher-level assets such as multimedia content, personal opinions, and news items. These resources can be *rivalrous*, meaning that consumption by one user may exclude others from consuming the same resource, or *non-rivalrous*, where the previous restriction does not apply, at least until *congestion* occurs. Storage space is an example of a rivalrous good, and any form of digital content is an example of a non-

rivalrous good because several users can consume the same digital content simply by replicating it. Whether or not congestion represents a problem depends at the layer one examines the system. For example, it is possible for file replication to cause congestion at the underlying transport layer.

Autonomous behavior In P2P systems, peer independence is an important design parameter. Peers may join and leave the system at any time. In addition, peers that are part of the system may dynamically tune their rates of contribution and consumption. System functionality does not rely on any one specific peer and the P2P system as a whole adapts to this dynamic behavior of its components. For this reason, P2P systems may perform well even in cases of random network outages and accidental node malfunctions.

Free-riding and altruism Free-riding is a common problem that P2P systems have to face. Peers tend not to contribute resources in order to minimize their own costs, which is a *rational* behavior. At the same time, free-riders benefit from the contributions of other peers. Altruism is the opposite force that may, at times, stabilize a P2P system in the presence of free-riders, but altruism can also cause more free-riding. In general, however, in order for cooperation to evolve and sustain in a P2P system, incentive techniques may be required.

Given the above definition of P2P systems, P2PWNC is a P2P system for the sharing of Internet access bandwidth over WLANs. One distinctive characteristic of the P2PWNC shared resource is that it is associated with a specific physical location. Under specific assumptions concerning the absence of congestion at the WLAN and Cable/DSL connection being used for backhaul connectivity, the P2PWNC resource can be thought of as non-rivalrous, and we can redefine the P2PWNC resource to be WLAN *coverage*. If the system is not overloaded, the WLAN coverage enjoyed by one consumer does not reduce the amount of WLAN coverage available to other prospective consumers. The main problem that the P2PWNC incentive techniques solve is how to encourage peers to provide additional WLAN coverage. P2PWNC coverage is an increasing function of the number of APs that are active, connected to the Internet, and support the *P2PWNC protocol* (see Chapter 7—Protocol and Implementation).

A P2PWNC system must respect peer autonomy and peer privacy (identity privacy, location privacy, and communication privacy—see Chapter 3 for definitions), and it must be an open system that anyone can join.

4.2 Overview of the P2PWNC scheme

P2PWNC works as follows. P2PWNC individual participants divide into *teams*, each team having a number of *members*.

Note, that the *teams* and not the individual participants are the *peers* in P2PWNC. In all that follows, for presentation clarity, both terms (*team* and *peer*) are used. However, the two terms are interchangeable.

Teams own and manage a number of APs (WLAN access points, connected to a network, for example the Internet, through DSL/Cable lines) at locations throughout an (urban) area. A team *consumes* whenever one of its members uses an AP of another team, and *contributes* whenever a member of another team uses an AP of this team ("using an AP" means "use the AP to access the Internet"). The objective of the P2PWNC incentive techniques is to encourage teams to match their consumption with an equal amount of contribution. "Consumption" refers to the volume of traffic (both egress and ingress) that a team member transfers through WLANs that belong to other teams, and "contribution" refers to the volume of Internet traffic that a team relays for members of other teams through APs that belong to this team. *Free-riding* teams that contribute less than they consume will obtain service of lower quality; and only short-term history is important: Teams must contribute continuously in order for their members to be able to consume continuously.

Members sign digital *receipts* when they consume service from another team. The receipts form a *receipt graph*, which is used as input to a *reciprocity algorithm* that identifies contributing teams using *network flow* techniques. Simulations show that this algorithm can sustain reciprocal cooperation. Receipts are distributed among multiple *team servers* with a *gossiping algorithm*. Two teams never exchange information over the Internet: Team servers are not accessible to people outside the team. Different teams interact only when a member is physically close to the AP of another team and requests its services.

The receipt graph may contain fake receipts—the result of *false trading*, usually accompanied by a *Sybil attack*. All member IDs and team IDs are unique public-private key pairs and the assumption is that it is computationally infeasible to break the digital signature scheme used to sign receipts and *member certificates*. Member certificates are signed by the

founder of each team. No Public Key Infrastructure is required, and teams do not know or keep track of the IDs of other teams. The system's *history* is represented by the receipt graph.

No *searching* for resources occurs in P2PWNC, unlike common P2P file-sharing systems: The members, when mobile, always request to access the P2PWNC AP that is closest to their current location. In addition, when granted access, members tunnel all their traffic to a trusted VPN gateway. Therefore, the security of the wireless link is not a problem and visited P2PWNC APs cannot eavesdrop on the foreign traffic they relay. The *receipt generation protocol* ensures that *hijacking* of the wireless session cannot occur.

The P2PWNC scheme is can be implemented directly on WLAN APs and WLAN clients. A custom P2PWNC implementation developed in the context of this dissertation proves this; it runs on the Linksys WRT54GS AP [30] and the QTEK 9100 WLAN-enabled cell phone [13].

P2PWNC systems are seen as a complement to 3G networks in metropolitan areas. The growth of WLAN and DSL/Cable deployments makes P2PWNC relevant. The P2PWNC scheme is straightforward to implement and it can achieve its basic goal, which is to stimulate the growth of WLAN deployments while respecting the autonomy and privacy of their owners.

4.3 P2PWNC distinctive characteristics

4.3.1 Location-based resource; no concept of search; contributor advantage

Unlike P2P file-sharing systems, the P2PWNC resource is associated with a specific location (the area covered by a specific P2PWNC AP). Prospective consumers need to be physically close to the contributing AP. The main assumption is that consuming peers do not *search* for an appropriate contributor; rather, they try to consume from whichever P2PWNC AP is closest at the time. Therefore, in P2PWNC, the prospective contributor has an advantage: his decision on whether and how much to cooperate is the only decision that affects the outcome of the prospective transaction. Unlike other P2P systems, the consumer is the only party in a P2PWNC transaction that needs to prove his "good standing" in the community to his contributor. (We can assume an extension here, however, to be used in *dense* WLAN environments, where search could play a role.)

4.3.2 Peer selfishness and rationality

P2PWNC peers are assumed to be selfish and rational. The participants adopt one of two roles during the system's operation: they can be *contributors* or *consumers*. Peers will try to benefit as much as possible while minimizing their costs. Because of this, incentive techniques that discourage selfish behavior and encourage cooperation are built into P2PWNC. The selfishness and rationality of peers in P2PWNC was not an afterthought but a requirement that constrained all the P2PWNC design decisions.

4.3.3 Lightweight incentive techniques

Another distinctive characteristic of P2PWNC is that the incentive techniques adopted are *lightweight*. The P2PWNC system encourages the sharing of Internet access bandwidth over WLANs so that pedestrians and other low-mobility users can access the Internet, for example, to place and receive voice- and video-phone calls or to access the Web. The idea is that users will use P2PWNC instead of resorting to the more expensive services of public 2G and 3G cellular carriers. The P2PWNC service is thus valuable, but its value must not be overestimated given the obvious alternatives. This means that the complexity of participating in a P2PWNC system and the overhead of the related incentive techniques must be low. Although this is difficult to quantify, one example of a lightweight incentive technique is to forego registration procedures that would involve an authority. Another is to minimize the need for additional hardware and software required to run P2PWNC—both in the peer's contributor and in the peer's consumer role.

4.3.4 Short-term history

Another P2PWNC requirement was that the incentive techniques should rely on short-term history only. The *history* of P2PWNC is represented by a set of *receipts*, that is, a set of records of prior transactions between peers (or, more accurately, between peer IDs). Every such record is associated with a timestamp, which encodes the time when the relevant transaction took place. These receipts represent *positive* reports, in the sense that they can only show the amount of service (measured in volume of relayed traffic) that the contributing peer contributed to the consuming peer—without any other information regarding, for example, service quality. Using these reports, the *reciprocity algorithm* (see Section 5.1) computes a *Subjective Reputation Metric* for the prospective consumer, and

according to this metric, the prospective contributor can decide whether or not the prospective consumer deserves to be served, and how much to be served. However, in order to encourage continuous contribution, the fact that a peer contributed resources some time in the past should not give him the right to exploit such past contributions continuously. By forgetting older receipts, continuous contribution is encouraged, and the practical problem of storing long-term history is also solved.

4.3.5 Free-ID regime

A distinctive characteristic of P2PWNC is its *free-ID regime*. This is a logical consequence of the full decentralization of a P2PWNC system. No authority exists to issue electronic IDs to peers and to control which peer is eligible to join and which is not. This is usually similar to the identity regime in P2P file-sharing communities: There, the peers choose their own ID when they first configure their P2P client. This ID is usually independent of their IP address.

There are many side effects to the free-ID regime. First, *recognition* in P2PWNC is difficult. In practice, when the AP of a contributing peer is set up with a given ID, this ID will not change regularly, and, consequently, this ID will be tied to a specific location. However, this location-to-ID binding is easy to change. Second, the free ID regime allows consuming peers to consume while constantly changing their IDs. This is the main reason why standard reciprocity-based strategies such as Tit-For-Tat (TFT) [57] would be ineffective in P2PWNC: the TFT strategy specifies that when a peer interacts with a peer he has never met before, he must first cooperate in the hope of forging a long-lasting cooperative relationship. It has been proven [72] that TFT will not do well in an environment with free IDs, and for obvious reasons: Rational consumers would constantly attempt to consume with a new ID in order to exploit the followers of the TFT strategy.

In addition, no *crypto-puzzles* are used in P2PWNC (see Chapter 2—Literature Review, and the KARMA system [60]). Such a defense against the creation of multiple IDs gives the perverse incentive to spend additional CPU power to solving crypto-puzzles in order to obtain an arbitrary number of IDs, sufficient to launch any Sybil attack. This is especially true in cases where the crypto-puzzles can be pre-computed. In P2PWNC, the model is simple: A P2PWNC peer can create an arbitrary number of P2PWNC IDs if it is in his interest.

4.3.6 Incentives for storing and sharing community history

Another distinctive characteristic of the P2PWNC incentive scheme compared to other proposed incentive schemes for fully self-organized P2P systems is the manner with which it deals with the issue of *accounting-layer incentives*. Many proposals for P2P systems (including Karma, EigenTrust, see Chapter 2—Literature Review) assume that it is possible to create a P2P overlay network in order to store community history using the same selfish peers that, at a higher-level, refuse to share their resources simply for the good of the P2P community. This assumption represents an easy solution to the problem of decentralized storage. This way, designers of P2P systems can first propose a design for a history-based reciprocity algorithm that works for an idealized centralized case, where a single server stores the entire history. Then, they can simply assume that the same history can be stored in a decentralized environment with the help of a *distributed hash table* [49, 50, 51, 52, 53, 54] or any other type of distributed overlay. This solves all the information dissemination problems. Such an assumption is not always justified. If peers can tamper with software, and because software code for automatic participation in overlay networks is still not part of common operating-system code, one cannot assume that peers will gladly store and share accounting information (such as records of prior transactions, scores and credit balances, and punishment histories) with other peers. Storing and sharing information must also be beneficial to the peers that store and share it.

It is interesting to note at this point the case of the NICE proposal [67], which is a work that also addresses the problem of accounting-level incentives. In the NICE proposal, however, even though peers are selfish at the "accounting layer," in that they only store accounting information that is in their direct interest to store—NICE *cookies* that show that peers were good cooperators in the past—they also freely *share* this information with other peers that request it, in order to assist in the completion of transactions, acting as third parties. The NICE peers have nothing to gain directly by acting as third parties and by helping other peers to learn more about the parties they transact with—such a functionality could be removed from the NICE software without affecting the remaining functionality of a NICE peer. In P2PWNC, the history-based reciprocity algorithm assumes that peers do not freely share accounting information; rather, P2PWNC peers store and share accounting information that is in their interest (see Section 5.2 for the details of the specific *gossiping algorithm* used in P2PWNC to disseminate history information in the community).

4.3.7 Full self-organization; no super peers

P2PWNC is a fully self-organized P2P system. No authorities that can punish or reward peers exist, and there are no certification authorities that certify peer IDs. In addition, there is no logically centralized bank (physically centralized or decentralized) that stores, for example, account balances for each peer. A micro-payment scheme cannot be adopted in P2PWNC because micro-payment systems require an authority to be online (even occasionally) in order to check for *double spending* [78]. There is no well-known public key for a P2PWNC system as a whole, and no well-known IP address of a server that new peers can connect to in search of information. No one issues peer IDs except the peers themselves and there is no authority that stores the history of the community.

A fully self-organized system also rules out super-peers. Super-peers are effectively a logically centralized authority, even if physically decentralized: All peers in P2PWNC are equals and there are no peers playing a special role in the system.

4.3.8 Practical system

It is unreasonable to claim that a system is practical before it is deployed in the real world. The implemented P2PWNC prototype (see Chapter 7) only proves that P2PWNC can be implemented using readily available technology. There are many additional real-world problems that must be solved before a P2PWNC system can be deployed, and they are outside the scope of this dissertation (see Chapter 8—Discussion and Future Work). However, the following characteristics make P2PWNC a potentially practical system to deploy.

First, minimal communication between peers is assumed. There are no Internet-based communications between P2PWNC peers; only when a peer visits another peer and requests its services (over the local WLAN) do different P2PWNC peers communicate. This keeps the P2PWNC protocol simple, and peers do not need to advertise IP addresses to other peers, a fact which makes peers less susceptible to Internet-based DoS attacks.

Second, P2PWNC does not rely on tamperproof hardware or software; the objective of Chapter 6 (Evaluation) is to show that there is no need to tamper with P2PWNC modules. Third, the proposed protocol for P2PWNC functionality (see Chapter 7) can be implemented by vendors of WLAN access points and WLAN clients at low cost, as the prototype that was implemented in the context of this work demonstrates.

4.4 A note on congestion and the cost of sharing

It is assumed that WLAN and DSL/Cable line congestion is not a problem. This becomes important in Chapter 6 (Evaluation), which presents a specific benefit-cost model for P2PWNC contributors and consumers. In practice, this assumption suggests that there will always be enough bandwidth for the users of P2PWNC to share, and that the major problem addressed by the P2PWNC incentive techniques is the lack of sufficient coverage. The assumption about lack of congestion allows the focus to be on the public good (more accurately, club good) aspects of the provisioning problem without emphasizing low-level WLAN-level details.

However, there is cost in sharing, and the assumption is that this cost is linear to the number of allowed visitors. This benefit-cost model is presented in Section 6.1.3 in more detail. In general, however, there are many indirect benefit and cost generators in P2P systems. The issue of congestion is discussed again in Chapter 8.

4.5 Teams: the P2PWNC peers

So far, "peers" represented the basic interacting elements in P2P systems. In the systems surveyed so far, a peer was usually an individual who, with the assistance of appropriate hardware and software modules, consumed from and contributed to a P2P system. However, as mentioned in Section 4.2, in P2PWNC the basic element of the P2PWNC system is the *team* and a P2PWNC system is composed of interacting teams. A P2PWNC team contributes to a P2PWNC system by operating WLAN APs that members of other teams can use, and consumes from the system by operating WLAN clients that request services from APs that belong to other teams.

The interacting P2PWNC entities are called teams because this also conveys the notion of a game being played (a game without a referee). The game involves opposing teams whose members are cooperating for the good of the team. All of the previous assumptions regarding selfishness and rationality apply to teams as a whole and not to individual players (who always care for the good of their team). Section 4.7 discusses again the assumption of cooperation within a team. For now, think of the standard P2PWNC team as a household, whose members collectively own and manage a single residential WLAN AP, which also functions as a P2PWNC AP; each member also owns a WLAN client.

The sections below look inside P2PWNC teams, discuss the roles of the individuals that compose a P2PWNC team (Section 4.5.1), and the software and hardware modules that

are necessary in order to support a P2PWNC team (Sections 4.5.2, 4.5.3, and 4.5.4). The P2PWNC protocol becomes relevant only when the interacting WLAN AP and WLAN client belong to different teams.

The concepts of a *P2PWNC AP* (Section 4.5.2) and *P2PWNC client* (Section 4.5.3) are introduced below. At the protocol level, the interactions between teams occur when a P2PWNC client accesses a (foreign) P2PWNC AP. Normally, the P2PWNC client would be part of a WLAN client such as a WLAN-enabled cell phone. The P2PWNC AP would be a WLAN AP with P2PWNC software embedded in its firmware, similar to how, in the custom P2PWNC implementation presented in Chapter 7, a P2PWNC AP is embedded in the firmware of the Linksys WRT54GS AP [30]. The P2PWNC client represents the P2PWNC peer in its consuming role and the P2PWNC AP represents the peer in its contributing role.

4.5.1 Founder, members, and member certificates

Each P2PWNC team has a *founder*. The role of this individual is to generate the team ID. This ID is a public-private key pair. The founder is required to keep the private key secret. This private key can then be used to sign *member certificates*. Member certificates are digitally signed documents that bind the team ID to a *member ID*. A member ID is a public-private key pair that is generated by each team member. The team member is supposed to keep his private key secret (even from members of the same team). By having a member certificate signed by the team founder (using the team's private key), the member is allowed to consume in the name of the team. Because it is easy to generate public-private key pairs, it is easy for anyone, in principle, to establish a P2PWNC team.

Figure 4.1 shows member certificates. Compared to a more traditional digital certificate, two things are missing from P2PWNC member certificates. First, there is no signature by a Trusted-Third Party; this is because P2PWNC does not rely on authorities. The only relationship that a member certificate encodes is the relationship between a specific member ID and a specific team ID. One can conceivably create numerous "fake" teams, each team with numerous fake members.

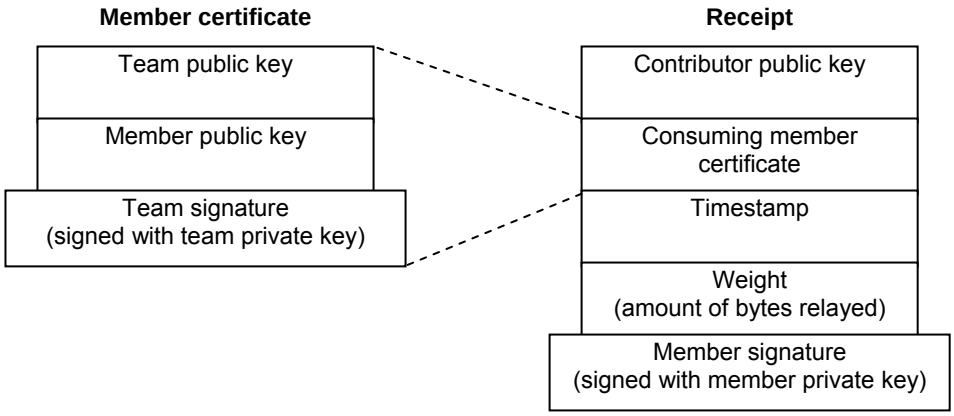

Figure 4.1 The format of P2PWNC member certificates (Section 4.5.1) and P2PWNC receipts (Section 4.6.1)

The second item missing from the member certificate is an expiration date. Traditionally, an expiration date is used to associate a specific duration with the trust relationship (the binding of the member ID to a specific team ID) that the certificate encodes. In the case of P2PWNC, because no commonly accepted certificate authorities exist, all that such an expired certificate would mean when interpreted from the perspective of other teams is that the particular member ID is no longer associated with the particular team ID. However, this would only complicate the strategic decisions available to contributing teams. To avoid complications, the member certificate is as simple as possible. In any case, expiration dates on certificates are not enough to solve the problem of a team-member relationship that ends abruptly: In practice, this is usually achieved by using explicit online Certificate Revocation Lists, a solution that would require a commonly accepted central server with the knowledge and the incentives required to make such a list available.

In the case a member *must* be expelled from a team, the following work-around is supported: the team founder must generate a new ID for the team and use the corresponding private key to sign certificates for all the members, save the excluded member. Because the P2PWNC relies on short-term history only, such a decision would represent only a temporary setback for a team. In any case, however, one of the functions of the team founder is to accept within his team, members who are not likely to be excluded in the future (such as family and household members). Section 4.7 discusses that the main reason of exclusion would be over-consumption by the specific member, which would cause the team as a whole not to be able to balance its consumption with its contribution.

4.5.2 P2PWNC AP

Having discussed a team's founder, members, and member certificates, the focus is now on the software and hardware modules that enable a team to *contribute* to the community. This functionality is abstracted inside a hardware and software combination called a *P2PWNC AP*. The easiest way to define the P2PWNC AP would be through the protocol commands that it can understand and handle (for this, see Chapter 7).

The APs that a team collectively owns and operates define the coverage that the team offers to the community. Team APs will probably be located in members' households (probably near windows, in order to increase the coverage they offer to passersby). Fortunately, the physics of electromagnetic propagation guarantee that APs can cover substantial areas surrounding a household, especially if specific antennas that boost their signal are used. These antennas can be omni-directional or directional, depending on the particular location of the household. For example, a household near a city square will preferentially choose to target the square with a sector antenna. The objective of team APs is to "earn" as many potential customers as possible, and in the process, raise the contribution level of the entire team.

Several options can be used as backhaul Internet connectivity for these APs; a popular solution will be DSL/Cable lines that connect to the Internet through commercial ISPs. If the objective of a P2PWNC system is to enable the use of VoIP within a metropolitan network, Internet connectivity may not be required at all.

Furthermore, the P2PWNC protocol requires that an AP that is part of the P2PWNC must use an SSID that makes it easy to detect it (see also Chapter 7). In particular, a P2PWNC AP must use an SSID starting with the string "P2PWNC-" A P2PWNC AP that does not use that loses potential consumers, and a non-P2PWNC AP that does that only exposes itself to unwanted P2PWNC connection requests. There is nothing an AP can gain by pretending to be part of P2PWNC: Clients, when offered Internet service, will use VPN tunnels to protect the confidentiality of their data from the visited APs that relay their data to the Internet.

In addition, there is no point in using IEEE 802.11 security schemes such as WEP [7] or WPA/802.1X [10] to protect the wireless link against eavesdroppers. These schemes assume that the users trust the AP; in the case of P2PWNC, the AP is itself just another potential eavesdropper, and VPN tunnels are the standard way to defeat such an attack by the visited AP.

4.5.3 P2PWNC client

"P2PWNC client" is the abstract term used to describe the device that a member of a P2PWNC team will use to consume the P2PWNC resource from other teams. It is likely that these devices will be WLAN-enabled cell phones, but any WLAN client can become a P2PWNC client in principle by installing appropriate P2PWNC client software.

Whenever a P2PWNC client is activated, and if a P2PWNC AP is nearby, the client may initiate the P2PWNC connection process. After a P2PWNC client connects to a P2PWNC AP, the P2PWNC client will tunnel all its Internet traffic through to a trusted VPN gateway.

4.5.4 P2PWNC team server

The most important component of a P2PWNC team is the *team server*. The team server's primary role is to run the *reciprocity algorithm* (see Section 5.1) and coordinate the APs of the team, in the sense that all team APs must first consult the team server before making a contribution decision. They do this through special P2PWNC protocol messages—QUER and QRSP—exchanged between the APs and the team server (see Section 7.1). At the end of a P2PWNC WLAN session, the AP that contributed service sends the receipt that corresponds to the completed P2PWNC transaction (WLAN session) to its team server for storage.

The team server is also responsible for coordinating the gossiping algorithm (see Section 5.2). The team server sends stored receipts to team members whenever they request them (in order for the P2PWNC clients to show them to foreign APs when gossiping—see Section 5.2). A design feature of the P2PWNC scheme is that the IP address of the team server need never be exposed outside the team. Team servers of different teams never talk to each other, and the risk of DoS attacks because of exposed IP addresses is lowered.

For practical reasons, the team servers also have finite capacity—an upper bound to the number of receipts they can store (see also below, Section 4.6.3).

4.6 Receipts

4.6.1 Format

A receipt is evidence that WLAN service was contributed and consumed. Receipts are signed by consuming team members whenever they use an AP of another team; receipts

are transferred to the contributing team during the WLAN session over the WLAN link. The *receipt generation protocol* is described in Section 4.6.2. This protocol ensures that none of the two parties (P2PWNC client/consumer or P2PWNC AP/contributor) is at a significant disadvantage during this exchange. Figure 4.1 shows the format of receipts. Receipts contain: (1) the public key of the contributing team, (2) the certificate of the consuming member, which contains the public key of the consuming team, (3) a timestamp noting the start time of the WLAN session, (4) a weight noting the volume of traffic the member transferred through the AP during the WLAN session, and (5) the member's signature, that is, a hash of the above fields encrypted with the consuming member's private key.

Two signatures that the receipts contain (the team signature on the member certificate and the member signature on the receipt) can be verified using information on the receipt itself (the team public key and the member public key). If both verifications succeed, this means that a team with a given ID (found on the member certificate) has authorized the member who signed the receipt to consume in the name of the team.

Receipts form a logical *receipt graph*. The vertices of this graph are team IDs and the weighted directed edges point from the consuming ID to the contributing ID. An edge's weight is equal to the volume of traffic the source has consumed from the target, that is, the edge weight is equal to the sum of the weights of the corresponding receipts. The direction of the edge thus denotes an "owes to" relation (see also Section 5.1.1).

Depending on the public-private key encryption scheme used, each receipt can range in size from 211 bytes to a few kilobytes. The P2PWNC protocol supports both RSA and ECC signatures of varying sizes (sizes which depend, in turn, on the length of the corresponding ECC or RSA keys). Sample signature verification and generation times for these two schemes are given in Section 7.2.

4.6.2 Receipt generation protocol

During a WLAN session, APs periodically request receipts from all foreign P2PWNC clients that are connected over the WLAN in order to account for the traffic relayed up to that point in the session. The interval between these requests is defined by the AP. These *receipt requests* (or RREQ messages, see also Section 7.1) contain the public key of the contributing team. Visiting consumers are then required to sign an appropriate receipt, which *must* contain the weight (volume of relayed traffic) the contributor is

currently measuring. As an important side effect, by receiving such a receipt, the AP can be sure that the consumer's session has not been hijacked because nobody else possesses the private key that corresponds to the member certificate that was presented initially, when the prospective consumer first connected to the P2PWNC AP (by issuing a CONN request, see also Section 7.1).

APs will only hold on to the last receipt in such a series. Every receipt is uniquely identified in the system by the {contributor public key, consumer certificate, timestamp} set of fields, so another receipt with different weight but the same timestamp (which always encodes the session start time) will be seen as the same. Thus, the AP has an incentive to keep the receipt with the biggest weight (see also Chapter 5—System Algorithms); the receipt with the biggest weight will be the last one in the series.

By requesting receipts periodically, none of the two parties is at a significant disadvantage. To guard against P2PWNC clients that attempt to avoid signing even the first receipt, the AP, in the beginning of a session, can ask for receipts at a higher rate, and may lower this rate only after the visiting member has proven trustworthy as far as receipt signing is concerned. This way the amount of unacknowledged relayed traffic can become arbitrarily small (of course, there is the practical concern of overloading both the WLAN client and the AP with numerous signature generations and verifications, respectively).

There is no concept of explicit disconnection in P2PWNC sessions: as soon as a visiting consumer refuses to sign a receipt (either because he finished his session or because he disagrees with the weight on the receipt), the session ends from the AP's point-of-view. The AP will store the last receipt from that session in its team server, and will block further consumer traffic by configuring the local firewall to stop relaying packets for the IP address used by the specific consumer.

Because the receipts also contain a timestamp that indicates the session start time, there needs to be loose time synchronization between the client and the AP, so that the client can verify that it is not signing a receipt for a time in the distant future or the distant past. This protects against APs that try to subvert P2PWNC's short-term history (see Section 4.3.4) by earning receipts that they can use in the future even if they stop contributing.

Figure 4.2 illustrates the receipt generation protocol.

4.6.3 Receipt repositories

After a WLAN session is over, the contributing team will store the newly generated receipt in its team server's *receipt repository*. Each AP is configured to be able to connect to its team server in order to do this. This way, all APs of the same team store the receipts they earn in the same receipt repository. When a receipt repository is full (the number of receipts it can store is a team-server parameter), the team server deletes the receipt with the oldest timestamp in order to store a new receipt.

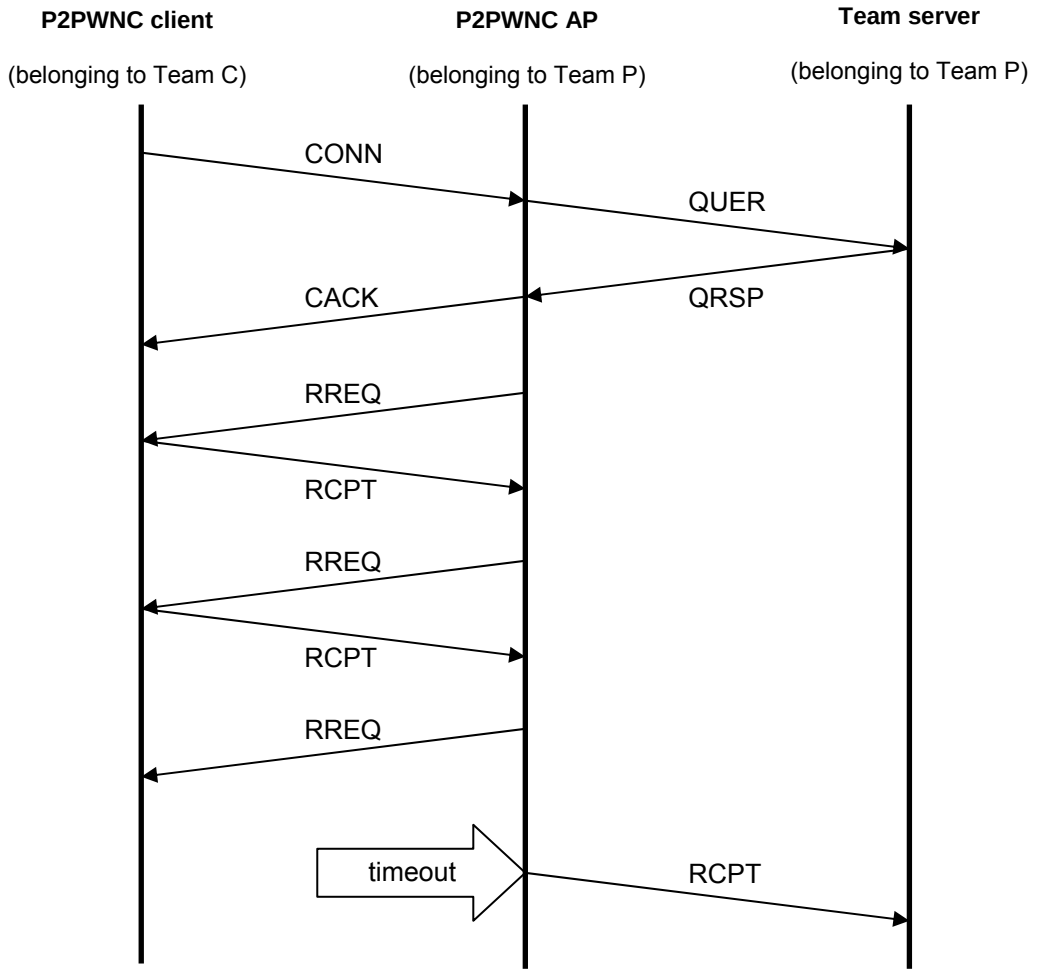

Figure 4.2 The receipt generation protocol. The actual messages are analyzed in Chapter 7. Here, after connecting and starting to use the service, the consumer signs a total of two receipts before he departs. The last receipt request is not answered, so, after a timeout, the P2PWNC AP stores the last receipt from the series of receipts in its team server.

A team's members may contact their team server in order to receive an *update* with the latest receipts from the receipt repository, which they will use when gossiping (see Section 5.2).

4.7 A note on teams

Figure 4.3 shows the components of a P2PWNC team. The details of how teams are formed are outside the scope of this dissertation (see also discussion in Chapter 8). One possible way to set up a team is to include in it all the members of a particular household. Such "household" teams will probably have only one P2PWNC AP.

The distinctive characteristic of teams as opposed to the P2PWNC community as a whole is that there is no anonymity within teams: the team founder accepts new members in the team by issuing member certificates signed with the team private key, which the founder holds. Although the real-world IDs do not appear in the member certificates, the founder can keep a list that maps a member public key to a member's real-world ID.

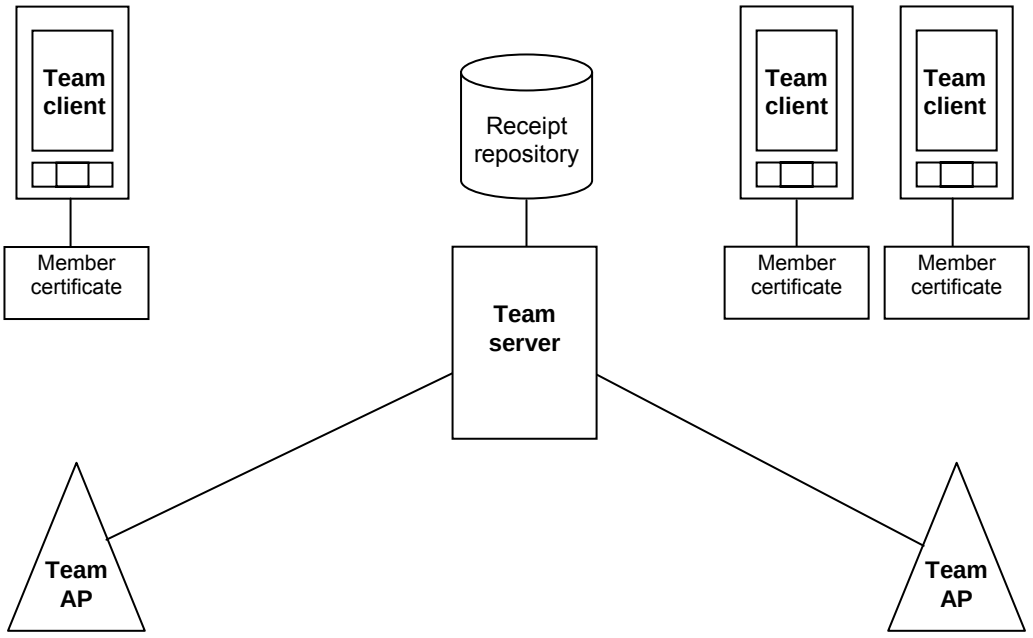

Figure 4.3 The components of a P2PWNC team/peer. A team uses APs to contribute to members of other teams, and clients to consume from P2PWNC APs of other teams. Each team/peer has exactly one team server that manages the receipt repository of the team.

Within a team, social control may work: Team members and the team founder can exert pressure on other team members to lower their individual consumption rate for example.

A reason to exclude a member from an existing team would be over-consumption by that individual member. If social control works, the member may relinquish his member certificate and never consume in the team's name again, or curtail his consumption rate. However, because individual team members will also be selfish, team members do not send the outgoing receipts that they sign (indicating consumption) to their team server. However, a team server is bound to found out about some of these receipts through the gossiping algorithm (see Section 5.2). Therefore, a team founder can look at the member certificates in the receipts and see if the percentage of the outgoing receipts of a team that come from this member exceed a threshold. If this member is not cooperative and is not willing to relinquish his member certificate, the team can create a new ID for itself (that is, generate a new public-private key pair), issue new member certificates for all the members save the excluded member, and start to advertise this new public key when contributing to the community.

The main advantage of using teams is that teams can include individuals who are living in areas where there are not many users to serve. Such individuals would gain little value from P2PWNC because the system enforces that peer consumption rate approximates peer contribution rate. If, however, these individuals could team with others that were willing to make up the difference at the team level, then the aforementioned individuals could find the system valuable. The consumption of these "peripheral" members would be balanced by the contribution of the "central" members that would on average contribute more than they consume because of location.

Another advantage is that, with teams, location privacy (untraceability) is enhanced. If teams own more than one WLAN AP, then a receipt by a foreign consumer indicating consumption is not enough for the community or for his team to find out where this consumption took place. Assume member *c*, a member of Team C, consuming from Team P. If Team C discovers the receipt that resulted from this transaction, it will know which member issued it but it cannot know the exact WLAN AP of Team P where the transaction took place (unless of course Team P cooperated and revealed the information, which we assume is impossible).

Chapter 5
System Algorithms

In this chapter, we will present the three algorithms that are central to the design of P2PWNC. We will evaluate these algorithms through simulation-based experiments in the chapter that follows (Chapter 6—Evaluation). In order of presentation, the three algorithms are: (1) the *reciprocity algorithm*, (2) the *gossiping algorithm*, and (3) the *bootstrap algorithm*. In our role as P2PWNC designers, we propose these three algorithms for use in deployed P2PWNC systems. Note that we have no way of preventing P2PWNC participants from using modified versions of these algorithms or using entirely new custom algorithms. The burden is on us, therefore, to show that, by adopting the algorithms that we present here, a participant will perform better (in a manner to be defined in the next chapter) than by adopting a more obvious alternative.

As system designers we are also concerned about the overall welfare of a P2PWNC community (its *social welfare*, see Chapter 3—Definitions). We will show in the next chapter that, if participants adopt the algorithms that we propose here, cooperation can evolve and sustain in P2PWNC communities. Naturally, however, there are many potential alternative algorithms. We claim, however, that our three algorithms represent a good trade-off between simplicity and effectiveness, and that the reciprocity algorithm in particular deals successfully with some of the attacks that selfish participants can launch. However, cooperation enforcement is like security engineering: there may still be attacks, not examined in this chapter and the next one, which can render our algorithms ineffective. We will outline such a potential attack in Chapter 8 (Discussion and Future Work).

5.1 Reciprocity algorithm

The reciprocity algorithm guides the cooperation decisions of a P2PWNC team who decides to adopt it. For the purposes of presenting and explaining the inner workings of this algorithm, we will simplify our model of a P2PWNC system. More specifically, we will ignore for now the existence of teams and we will view the P2PWNC system simply as a P2P system where *peers*, and not teams, are the entities that interact. (Remember that a peer is, in actuality, a team, consisting of (1) individual members, (2) the P2WPNC clients that these members use to consume service from other teams, and (3) the P2PWNC APs

that the members use to contribute service to members of other teams.) We can therefore model P2PWNC peers in the abstract as entities, each with a unique ID (the team public key), which consume and contribute the P2PWNC resource (WLAN and Internet bandwidth). This is equivalent to saying that we will be examining teams with exactly one P2PWNC client and exactly one P2PWNC AP per team.

A natural consequence of the above simplification is that we can also use a simplified model for P2PWNC receipts. We will therefore disregard for now that receipts contain P2PWNC member certificates (see Section 4.6.1) and we will focus on the fact that P2PWNC receipts are records of a (completed) WLAN session, and we will assume that each receipt contains exactly the following four information fields:

(1) The ID (in reality, the team public key) of the consuming peer.
(2) The ID (in reality, the team public key) of the contributing peer.
(3) A timestamp, noting the start time of the corresponding WLAN session.
(4) A weight, noting the total amount of bytes relayed during the WLAN session.

We will also disregard for now that the IDs mentioned above are implemented using public keys of a specific length and use specific public key algorithms. We only need to assume that each peer ID (public key) in a P2PWNC system is unique.

We will also assume that each receipt can be identified uniquely in the system, using the tuple that comprises the following fields: {consuming peer ID, contributing peer ID, timestamp}. This is less realistic than adopting as unique receipt ID the tuple of fields that we mentioned in the previous chapter, which also contained the member certificate. This is because, without the member certificate, there is a non-zero probability that two members from the same team will start consuming from the same foreign team at precisely the same time. (Note that the receipt timestamp describes a session's *start* time.) However, this is not important for the analysis that follows.

5.1.1 The receipt graph

Using the simplified model for receipts we presented above, we can now see that a set of receipts defines a logical *receipt graph* **G**, with the following characteristics:

(1) The vertices in **G** represent peers (more precisely, peer IDs).

(2) **G** is a *directed* graph. A directed edge of the type $C \rightarrow P$ exists in **G** if the source vertex, Peer C, has consumed at least once from the destination vertex, Peer P.

(3) **G** is a *weighted* graph. The weight of the $C \rightarrow P$ edge is equal to the sum of the weights of the corresponding receipts. By "corresponding receipts," we refer to all the receipts that were issued in the system (from its inception) that show Peer C as the consuming peer and Peer P as the contributing peer.

As aid for the analysis that follows, we can think of the direction of the edges to describe an "owes to" relationship. Edges thus represent a *direct debt* that the source peer owes to the target peer. For example, assuming Peer C had consumed service from Peer P exactly once, and assuming the weight on the corresponding receipt was equal to 50,000 bytes, then **G** would contain a $C \rightarrow P$ edge, with a weight equal to 50,000. If Peer C had also consumed service from Peer P on another occasion, and a second receipt for 75,000 bytes was signed, then the edge $C \rightarrow P$ in **G** would have a weight equal to 125,000 (50,000 + 75,000). Note that for simplicity we always assume that we measure the volume of relayed traffic in bytes. This way we can have the weights on the graph edges be dimensionless, silently assuming that we are always discussing relayed bytes.

So far, we have not discussed the details of how the graph is realized. The details of the corresponding decentralized receipt storage subsystem (which relies on the receipt repositories found on team servers, as well as on the gossiping algorithm for receipt dissemination) are unimportant for the analysis of the reciprocity algorithm. We describe the actual realization of graph **G** and the internals of the gossiping algorithm in Section 5.2.

5.1.2 The meaning of cooperation in P2PWNC

For the following analysis, whenever we state "Peer P cooperates with Peer C" we mean that Peer P contributes an amount of P2PWNC/WLAN service to Peer C; that is, Peer P allows Peer C to access the Internet through a WLAN AP that belongs to Peer P. The result of such a cooperative action would be the eventual generation of a new $C \rightarrow P$ receipt, with a weight equal to the amount of traffic that P relayed for C during the WLAN session (see also the *receipt generation protocol* in Section 4.6.2, and the way we handle consumers who may avoid signing receipts). This receipt would then be included in **G**. This would result either in the creation of a new $C \rightarrow P$ edge if none existed before, or it would result in the increase of the weight of an existing $C \rightarrow P$ edge according to the rule that we

specified above. (The rule states that the weights of the edges in **G** must equal the sum of the weights of the corresponding receipts.)

The high-level objective of the reciprocity algorithm is to allow cooperation to evolve and sustain in a P2PWNC system. At the same time, it must be in the interest of selfish and rational peers to follow the reciprocity algorithm. We will show how our proposed algorithm is able to achieve these two distinct objectives in the following chapter (Chapter 6—Evaluation).

The idea behind the reciprocity algorithm is to use the threat of exclusion and the threat of offering service of reduced quality as an incentive to encourage cooperation by cooperating only with known cooperators. One obvious problem then is how to distinguish cooperators from non-cooperators.

We will introduce a metric, which we call the *Subjective Reputation Metric* (SRM). Its values range from zero to one, inclusive. Assume a prospective consumer, Peer C, that requests service from a prospective contributor, Peer P. Peer P will first compute the SRM for Peer C by examining the receipt graph. The closer SRM is to one, the more Peer P "owes to" Peer C. The closer this value is to zero, the less Peer P "owes to" Peer C. We will see in Section 5.1.4 that this metric cannot, and does not, measure a peer's contribution objectively—a peer's level of contribution appears different depending on the peer that examines it.

5.1.3 Reciprocity algorithm requirements

According to the system model that we presented in the previous chapter, there are certain requirements that the design of the reciprocity algorithm must take into account. We revisit below some of the system requirements that we presented in Section 4.3, which are directly relevant to the design of the reciprocity algorithm.

First, the reciprocity algorithm represents the basis of our proposed incentive technique for stimulating participation in P2PWNC systems. As such, the algorithm should be practical (see Section 4.3.3). We will not attempt to quantify this. However, we claim that the overall cost to the individual peer of running a particular reciprocity algorithm must be smaller than the value of the resource that the system as whole is designed to provide to the same individual peer. We mentioned previously that the value of the P2PWNC resource (wireless Internet bandwidth for pedestrians and low-mobility users) must not be over-estimated. A complex algorithm that is difficult to implement and understand could drive

peers away from the system in search of simpler alternatives, especially if these alternatives cost less in *user attention* [79]. An obvious alternative other than participating in a P2PWNC system is for users to rely on their existing 2G/3G cellular service.

Second, the reciprocity algorithm must function in a completely decentralized environment without super-peers (see Section 4.3.7). This is where the gossiping algorithm (Section 5.2) comes in: It provides the needed history input to the reciprocity algorithm, even though no central server or overlay network for storing this history exists.

Third, the algorithm must rely, for practical reasons, on short-term history only (see Section 4.3.4). The logical graph that we described before is not only distributed across multiple peers (see also Section 5.2—the gossiping algorithm) but also these peers have only a finite amount of memory. Thus, portions of the graph are constantly removed from its "stored" version as new edges (and even vertices) are added. None of the receipts in a P2PWNC system is stored forever; rather, older receipts are being discarded continuously. As we already mentioned, however, an algorithm that "conveniently" forgets older receipts also has the desirable side effect of encouraging continuous contribution.

Fourth, we assume that prospective consumers cannot trace the lack of P2PWNC/ WLAN service in a particular area back to a specific contributor who *could* have had contributed service but did not. Thus, they cannot punish him for not contributing. In that respect, the reciprocity algorithm must be able to function only with the positive reports—the receipts—that are contained in $\mathbf{G}$, many of which could be fake, the result of *false trading* (see Chapter 3 for a definition).

Fifth, we already mentioned in Section 4.3.1 that it is only the prospective contributor's decision that makes a difference as to the outcome of a P2PWNC transaction—the prospective consumer has no choice in the matter: We assume that the consumer simply requests service from a P2PWNC AP. Consequently, the prospective consumer does not examine the contributor's contribution or consumption history: Only the prospective consumer must prove to the contributor his good standing in the community. Therefore, only the prospective contributor executes the reciprocity algorithm.

Finally, we assume that the receipt graph is the only source of history information for the peers. We assume peers do not keep any additional information regarding other peers. They do not know the IDs of other peers, they do not communicate out-of-band, and they do not remember any information regarding other peers, except for the contents of the receipt graph. For example, peers do not maintain lists of known "friends."

5.1.4 Receipt graph security

As a further aid to our analysis of the reciprocity algorithm, we will examine the ways in which the attackers can affect the shape of the logical receipt graph. Because no authority oversees P2PWNC transactions, and because of the P2PWNC free-ID regime, attackers are free to engage in egregious false trading. They can create an arbitrary number of fake edges, with arbitrary weights, that do not correspond to real P2PWNC transactions (that is, they do not correspond to WLAN sessions that actually occurred). To do that, attackers must be in control of an edge's source vertex, that is, they must possess the private signing key that corresponds to the public key, which represents the ID of that vertex. They are able to control it if they created the source vertex themselves; and they are able to create a source vertex because each attacker can generate an arbitrary number of Sybil IDs in a Sybil attack. These IDs can exist in parallel with his real ID, or the attacker may possess no real ID at all. By "real ID," we refer to the ID that is associated with the P2PWNC AP, if any, that belongs to the attacker. That is, the real ID would be the ID that appears when (and if) the attacker actually contributes service to other P2PWNC peers, and requests for the generation of receipts that contain this ID as the contributing ID.

On the other hand, attackers cannot create edges starting from vertices that are not under the attacker's control, and they cannot change the weight of an existing edge nor delete an existing edge.

5.1.5 The inappropriateness of the Tit-For-Tat strategy

One way to stimulate cooperation in P2P communities is to exclude free-riders from consuming the community good. A basic reciprocity algorithm would specify something like the following: "you must contribute service if the requesting peer contributed service to you in the past, and you must not contribute service if the peer did not contribute service to you in the past". This is exactly what the Tit-For-Tat (TFT) algorithm [57] specifies. However, the TFT algorithm also specifies what must happen in case a peer comes across a peer whom he has never met before in the past—this is important, in order to solve the obvious bootstrapping problem. In such a case, the TFT algorithm specifies that the prospective contributor must cooperate with the prospective consumer in the hope of forging a long-lasting cooperative relationship. However, in an electronically mediated community with a free-ID regime, TFT is susceptible to a trivial

attack. A peer that follows the TFT algorithm can be continuously exploited by free-riders who constantly acquire new IDs (as shown also in [72]). These peers will be able to avoid the costs of contributing, while exploiting the naivety of the TFT algorithm, leading the community to reduced levels of cooperation because free-riders would be able to persist, having no incentive to change their strategy.

The second disadvantage of the TFT algorithm is that it relies on direct debt only. This would mean that a peer would need to be indebted directly to another peer in order to assist him. Using the results of [71], the idea behind our reciprocity algorithm is to rely on *cyclical reciprocity*: Peer P can cooperate with a Peer C that had cooperated in the past, not directly with Peer P, but with other peers that had directly (or indirectly) cooperated with Peer P. This type of reciprocity can scale to larger populations, as shown in [71].

A natural way to aggregate all the direct and indirect debt paths that connect two different peers on the receipt graph is by using the *maximum flow* graph algorithm (see also [71], and the analysis in Section 2.4). We describe how in the following section.

5.1.6 Maximum flow on the receipt graph and its meaning

The cyclical reciprocity algorithm relies on *maximum flow* (from now on referred to as *maxflow*), a graph theoretic algorithm that measures the amount of flow that can pass from a source vertex to a destination vertex of a weighted graph. Here, we analyze the benefits of such a maxflow-based reciprocity algorithm, and we also refer to [71], where a similar algorithm for encouraging cooperation was first proposed. In the next section, we will discuss a specific weakness of the maxflow-based algorithm presented in [71], and we will propose an additional heuristic that addresses this weakness.

Let us begin by using a basic example to show how a heuristic such as maxflow can assist in encouraging cooperation and in excluding free-riders from P2PWNC. In Figure 5.1, a prospective contributor, Peer P, must decide whether to contribute to a prospective consumer, Peer C. According to the graph that is available to Peer P, Peer C appears to be at the root of a tree of several receipts, all of which directly or indirectly point to Peer C, thereby signifying that Peer C contributed service in the past, and is therefore a good contributor that deserves to be served. But is he?

From Peer P's perspective, there is no guarantee that this entire tree of receipts is not the result of a Sybil attack on the part of Peer C. We claim that *only* if Peer P detects himself somewhere in that tree in the role of consumer can he be sure that at least one of the

receipts in this tree is real—and that would be his own outgoing receipt. (This is because we assume that peers do not trust the IDs of other peers, which could all be Sybil IDs of Peer C.) Peer P can then be sure that he owes, directly or indirectly, service to Peer C.

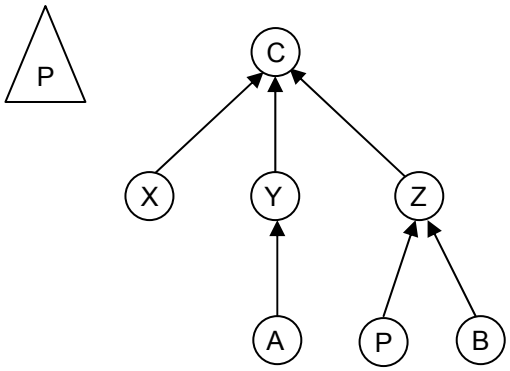

Figure 5.1 Peer C visits the access point of Peer P and requests service. Peer P, according to his view of the receipt graph, sees Peer C being the root of a tree that also contains Peer P. Peer P therefore owes service to Peer C (here, indirectly, via Peer Z). If Peer P did not detect himself, Peer C could have fabricated the entire tree (see also Section 5.1.4 on the security of the receipt graph).

How much service does Peer P owe exactly though? In order to avoid simple attacks by Peer C that attempt to present larger weights in some of the receipts in the tree, P's debt to C must be bounded by the only receipt that Peer P can trust to be real: his own.

For the same reason, any bottleneck that appears in the path from Peer P to Peer C must be taken into account. The debt should be bounded by any receipt with smaller weight that appears towards the root of the tree. For example, if P owes 1000 Kbytes to Z (that is, has signed receipts with a total sum equal to 1000 Kbytes), and Z owes 10 Kbytes to C, then P owes indirectly only 10 Kbytes to C.

One way to aggregate these paths of debt is to use the maxflow algorithm in a way similar to [71] (as presented in Section 2.4). That is, the maxflow result from P to C indicates the *total* indirect and direct debt that P owes to C. Following the example from [71], we can show that under maxflow, a Peer C that performs the Sybil attack can never appear to Peer P as contributor, no matter how much false trading he engages in. If no ID contributes service to the community, no matter the size of this *clique*, there will be no flow from Peer P to any ID in the clique, and the maxflow from Peer P to any ID in the clique (that is, the indirect debt) will be zero. See also Figure 2.1.

Below we will define the r_1 component of our Subjective Reputation Metric, which is based on the decision function of [71].

$$r_1 = \min\left(\frac{mf(P \to C)}{mf(C \to P)},\ 1\right) \tag{5.1}$$

where $mf(P \to C)$ stands for the result of maxflow from vertex P to vertex C. The r_1 component is an indicator of *how much more* Peer P owes to Peer C compared to what Peer C owes to Peer P (both indirectly and directly).

In [71], the heuristic proposed is for Peer P to cooperate with Peer C with a probability that equals r_1. Simulations in [71] show that a community adopting this algorithm can sustain high-levels of cooperation. We adapt this heuristic, and extend it by using another metric that we define below, which is designed to defend against a sophisticated attack.

5.1.7 Generalized maximum flow and its meaning

We have introduced a new metric, which we call GMF. GMF was inspired by the class of *generalized maximum flow* graph theoretic algorithms [80], hence its name. GMF captures the "directness of debt" on the graph. To calculate GMF from a source vertex to a target vertex on the graph, one executes a standard maximum flow algorithm first, in order to identify all the component flow paths that contribute to the result of maxflow. However, the result of GMF will in general be different from the result of the equivalent maxflow algorithm. This is because GMF *discounts* the value of flow as we move away from the source vertex. More specifically, for all the component flow paths that contribute to the result of maxflow, we multiply each component by $2^{(1-k)}$, where k is equal to the hop count in vertices that we need to traverse to get from the source vertex to the target vertex. If the GMF result equals the maxflow, then all the debt from P to C is direct: C is only one hop away on the graph. For two example GMF computations, see Figure 5.2.

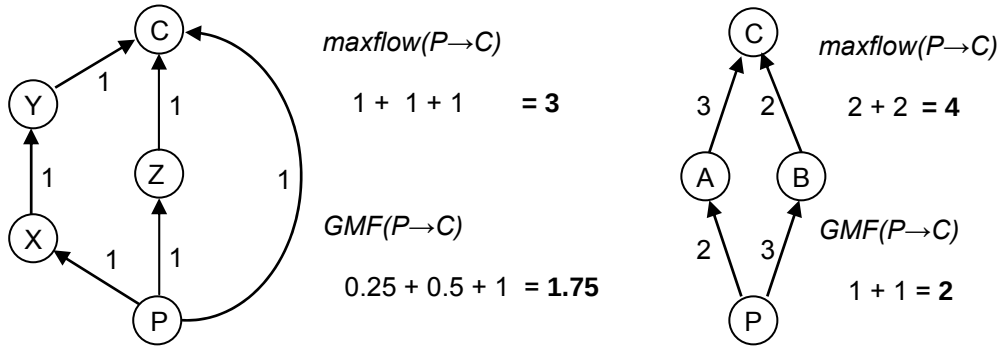

Figure 5.2 In the two graphs above we calculate both the maxflow and the GMF from P to C. Edges are weighted. After isolating the component flows that make up the maxflow result, GMF discounts each one exponentially, depending on the length of the path that the flow traverses. Thus, GMF is a measure of the directness of debt. If all debt were direct (one-hop away), the GMF result would equal the maxflow result. In the example above, the left-most flow P→X→Y→C that has a value of 1 is multiplied by $2^{(1-k)}$ with $k = 3$. See also Section 5.1.7.

We introduce GMF for the following reason. Assume we only used the heuristic of [71] (see also Section 2.4), that is, peers cooperated with probability r_1 (see Equation 5.1). Let then C be a consuming peer. However, instead of consuming using ID C, each time C consumes, he uses a fresh ID C_i, $i > 0$, which C never reuses. This Sybil attack does not benefit C, because $mf(\cdot \rightarrow C_i)$ is zero for all i (because there are no receipts pointing *to* the fresh ID). Therefore, P would not cooperate with C_i according to r_1.

However, if C falsely trades with the new ID and creates a new $C \rightarrow C_i$ edge, with weight w, then, if

$$w \geq mf(P \rightarrow C)$$

we have

$$mf(P \rightarrow C_i) = mf(P \rightarrow C)$$

that is, by the definition of maxflow, all the flow from Peer P that reaches Peer C will also reach C_i if the $C \rightarrow C_i$ edge weight is enough to carry it. In addition, $mf(C_i \rightarrow P) = 0$ (because the C_i ID is fresh and therefore has no outgoing receipts).

Effectively, this way, ID C_i earns all the "contribution reputation" of ID C, but none of its "consumption reputation." Note that ID C above *will* indirectly owe more to others as a result of these Sybil attacks. However, because the members of Team C would appear with a fresh team ID C_i each time, the denominator of the ratio in Equation 5.1 will always equal zero, and the decision function of [71] would decide that the probability to

cooperate is always one, as long as the numerator is positive. That is, if Peer P owed even the slightest indirect or direct debt to Peer C, he would always cooperate. This was not the design requirement for the heuristic in [71], which attempted to balance the debt from Peer P to Peer C with the debt from Peer C to Peer P. However, with the Sybil attack we described above, the second debt can always be zero. Figure 5.3 shows this attack.

The meaning of GMF is to detect the peers who perform this attack. GMF to such peers will be consistently less (half) than what it would have been had they not conducted this attack. See also Figure 5.4.

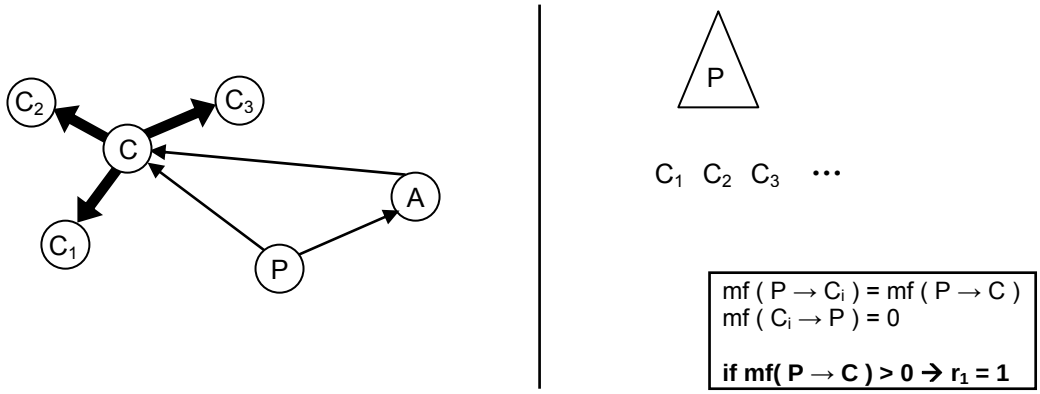

Figure 5.3 Peer C visits Peer P not as "C" but using fresh ID C_i. By using a fake receipt of sufficient weight, Peer C can pass all his "contribution reputation" to C_i. Because C_i is fresh, it has no "consumption reputation." In such a case, component r_1 always equals 1 if P owes a positive debt to C.

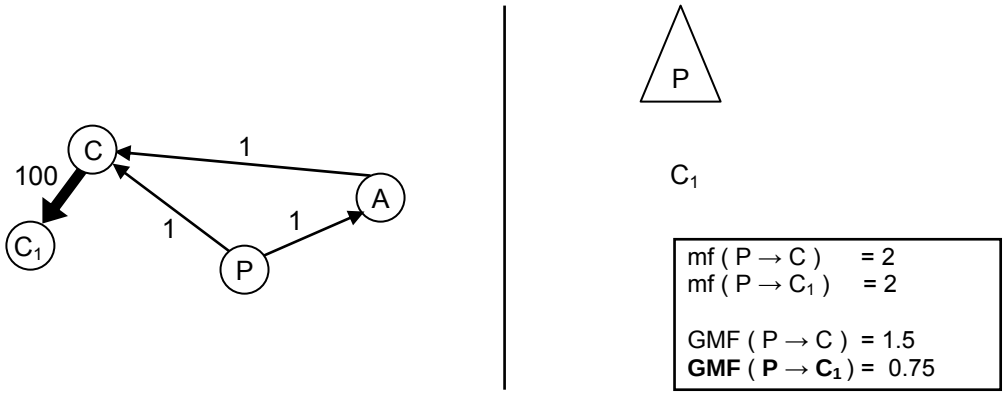

Figure 5.4 If Peer C performs the Sybil attack and pushes his "contribution reputation" one hop away to ID C_1, the GMF from P to C_1 will be half the GMF C *would* have gotten had he consumed as "C."

5.1.8 Formulas

The input to the reciprocity algorithm is the receipt graph and the ID of the prospective consumer, Peer C. The algorithm is run by the prospective contributor, Peer P, whenever a prospective consumer requests service. The output of the algorithm is a value between zero and one, inclusive, which we call the *Subjective Reputation Metric* of Peer C "in the eyes" of Peer P, and we denote $SRM_{P\rightarrow C}$ or simply SRM, as the two peers will normally be clear from context.

To compute $SRM_{P\rightarrow C}$, Peer P first needs to compute its two components, r_1 and r_2. We have already introduced the r_1 component:

$$r_1 = \min\left(\frac{mf(P \rightarrow C)}{mf(C \rightarrow P)},\ 1\right) \tag{5.2}$$

The second component, r_2, is

$$r_2 = \frac{GMF(P \rightarrow C)}{gmf_{avg}} \tag{5.3}$$

If gmf_{avg} equals zero, we set $r_2 = 1$.

gmf_{avg} is the average GMF that Peer P sees in the community. It is updated according to the formula below every time Peer P computes a new GMF result, GMF_{new}.

$$gmf_{avg} \leftarrow gmf_{avg}\ a\ +\ GMF_{new}\ (1 - a) \tag{5.4}$$

where a is a constant (we use $a = 0.75$ in the experiments). Finally, the $SRM_{P\rightarrow C}$ is

$$SRM_{P\rightarrow C} = \min\ (1, r_1\ r_2) \tag{5.5}$$

We introduce the second component in the SRM because of the possibility of the attack that we described above. gmf_{avg} is computed locally by each peer. At the start of peer's lifetime, it is set to zero, and it is only updated each time a positive GMF result is calculated by the peer (in order not to allow the free-riders that would be detected this way to influence the average).

The SRM metric will be evaluated in Chapter 6, where it will first be translated to an amount of *benefit* that would be contributed to the requesting peer. Also, the r_2 compo-

nent of the SRM will be evaluated through an attack that the r_1 component alone cannot handle. Both components are important: the first because it balances the peer's consumption actions with its contribution actions, and the second because it allows the detection (under the specific mobility models we will examine in Chapter 6) of peers who perform the attack we described above in an attempt to obtain the maximum SRM. Other additional implications of that attack are also discussed in Chapter 6.

5.2 Gossiping algorithm

5.2.1 Introduction

So far, we used the receipt graph as input to the reciprocity algorithm, but we did not discuss how it is realized. A P2PWNC system is fully decentralized, with no authority that can store the entire history of the community. Moreover, the peers do not communicate with each other over the Internet.

A peer in P2PWNC is a team, and each team has exactly one team server and one receipt repository, located at the team server. This receipt repository has a finite size—a maximum number of receipts that it can store. The team server is responsible for running the reciprocity algorithm whenever one of the team APs request it. Within such a request, a team AP supplies to the team server the ID of the prospective consumer (that is, the public key of his team); then, the team server computes the SRM for that peer (the Subjective Reputation Metric) and communicates it to the AP, which acts accordingly. At the end of a WLAN session, the AP sends to the team server the final receipt from the series of receipts that were generated in accordance with the receipt generation protocol (see Section 4.6.2). This receipt will be stored in the receipt repository of the team with other receipts. If the receipt repository has reached the limit of receipts it can store, the team server will remove the oldest receipt from the repository, according to timestamp.

If we did not add any further functionality to this system, two things would happen. Let us focus on Peer P: (1) the receipt repository of Peer P would only contain receipts that show Peer P as the contributor in transactions; (2) the value of SRM that is computed on such a partial view of the receipt graph—a graph that contains only edges that point directly to Peer P—will always be zero: no outgoing edges start from Peer P, so both the maxflow algorithm and the GMF algorithm, in the numerators of the r_1 and r_2 components respectively, will return zero.

A first step to make to address this problem would be to require that the consuming members of Team/Peer P must report the outgoing receipts that they sign to their team server. This way, a peer/team can become aware of its consumption actions, and its receipt repository would also contain receipts that show Peer P as the consumer in the transaction; that is, P's view of the graph would now also contain outgoing edges starting from vertex P. However, for reasons that we have already discussed in Section 4.7, this solution must be rejected—there is no straightforward way to force team members to report their consumption back to their team server and, according to the team-internal dynamics that we discussed in Section 4.7, rational team members would try to hide their consumption actions from their team server. This is one reason why the gossiping algorithm that we propose becomes useful.

5.2.2 Phase 1: Update

According to our gossiping algorithm, prospective consumers carry with them, in *mobile* receipt repositories (located on their P2PWNC clients), a part of their team server's receipt repository. More specifically, P2PWNC clients can periodically request an *update* with the latest receipts from their team server, which they then store in their mobile receipt repositories. Each team server, upon receiving an update request, sends some of the most recent receipts from the receipt repository to the team member who requests it. The P2PWNC clients can request for an update periodically, for example, every time they access the Internet through the P2PWNC system or through another system like 2G/3G cellular. Alternatively, they can update their mobile repositories every time they find themselves in the same local network as their team server (for teams that are organized around a household, this could happen as often as once per day). In the next chapter (Chapter 6—Evaluation), we will assume that the P2PWNC clients are constantly synchronized with their team server, always storing a fixed number of the most recent receipts. Because a receipt can be as small as 211 bytes (see Chapter 7), a cell phone-based P2PWNC client can store thousands of receipts in its mobile repository.

5.2.3 Phase 2: Merge

The update procedure above represents the first phase of the gossiping algorithm. The second phase involves the P2PWNC client showing the receipts from its mobile repository to the prospective contributor when the client requests service. Assume Peer C (that is,

a member of team C) is requesting service from Peer P (that is, an AP belonging to team P). At this point, Peer C has an incentive to show receipts from Peer C's repository to Peer P: These receipts, originating from Peer C's team server, will also contain receipts earned by Peer C that show Peer C being the contributor in transactions. If Peer P were to consider these receipts, and provide them as additional input to the reciprocity algorithm, then these receipts would only increase the numerators of both the r_1 and the r_2 components. Both the maxflow result as well as the GMF result for the flow from Peer P to Peer C can only increase if additional receipts that point *to* Peer C appear in Peer P's local view of the graph.

This way, not only does Peer C potentially increase the result of the reciprocity algorithm in his favor; also, P2PWNC receipts are disseminated through the system. Peer P will take the receipts that Peer C showed him, and *merge* them in his receipt repository. Our algorithm specifies that this must happen before the execution of the reciprocity algorithm. The standard rule concerning receipt replacement according to timestamp applies: Whenever a new receipt is inserted in the team server's repository, if the repository is full, then, the oldest receipt must be removed. Effectively, this means that Peer P will not merge in his receipt repository a receipt with a timestamp that is older than the oldest receipt timestamp in his repository—which, effectively, defines a *time horizon* for Peer P.

Now, following the same update procedure, P's team server will eventually send to P's members a fixed number of the newest receipts from Peer P's receipt repository—some of these receipts will have arrived through the same gossiping procedure. P's members, in turn, will show these receipts to the P2PWNC APs that they visit, furthering their dissemination through the system.

In addition, through this gossiping algorithm, each peer eventually learns about its own outgoing receipts (consumption actions) without requiring that its own members report these actions to their team server. Instead, these receipts arrive at the team server as a byproduct of the gossiping algorithm. Let us examine a basic example to illustrate how. Assume Peer C requests service from Peer P. Assume also that a member of team P was recently consuming from C. Then, the P2PWNC client from Peer C will have a receipt of the type $P \rightarrow C$ in its mobile receipt repository. Peer C will show this receipt to Peer P according to the gossiping algorithm, and Peer P therefore learns of one of its recent consumption actions. The signature on the receipt and the member certificate contained in the receipt are enough to prove that it was indeed a member of team P that signed this receipt.

Let us examine a second, slightly more involved example. Assume Peer C requests service from Peer P. Assume also that a member of team P was recently consuming from Peer A. Assume also that Peer A was recently consuming from Peer C. Then, it is possible that the client from Peer C will have in its mobile receipt repository both an $A \rightarrow C$ *and* a $P \rightarrow A$ receipt (because Peer A, before consuming from Peer C, showed the $A \rightarrow C$ receipt to Peer C).

The two phases of the gossiping algorithm can be seen in Figure 5.5.

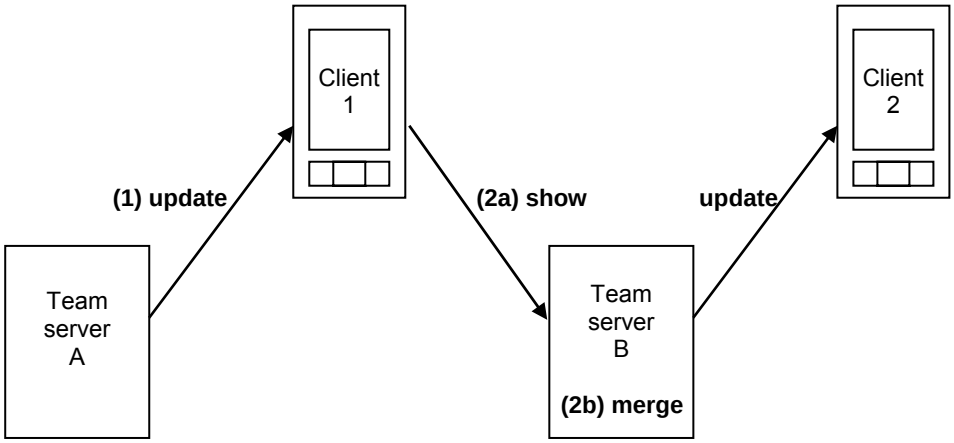

Figure 5.5 The two phases of the gossiping algorithm. Client 1, member of Team A, *updates* his mobile repository with receipts from his team server. When Client 1 requests service from Team B, he *shows* these receipts and Team B *merges* them with its repository. Then, Team B updates its clients with (some of) these receipts.

5.2.4 Gossiping and incentives

We note the following special case in the gossiping algorithm. Although we established that peers show the most recent receipts from their receipt repositories to their prospective contributors, *they never show their own outgoing receipts* (which they can learn through the same gossiping process). This is because, in the example above, if Peer C were to show to Peer P any receipts of the form $C \rightarrow X$, this could only serve to potentially increase the result of the denominator of component r_1—and Peer C is selfish and rational and does not want that. Therefore, an amendment to the gossiping algorithm is that when members issue an update request to their team server requesting a fixed number of the most recent receipts, their team server will never send them receipts that show their team as being the consumer. Teams have an incentive to hide these consumption actions from the gossiping algorithm (and an incentive to show contribution actions).

It is interesting, at this point, to consider if teams have an incentive to show (to their prospective contributors) receipts in which they do not appear at all—such as the $P \rightarrow A$ receipt that Peer C showed to Peer P in the example above. In this particular case (with the $P \rightarrow A$ receipt), this receipt can only help increase the numerator of both components r_1 and r_2 that Peer P will compute. In the general case, however, and assuming that the prospective consumer does not know the ID of their prospective contributor (which they do not; according to the P2PWNC protocol, the gossiping algorithm runs before the first receipt request message—see also Chapter 7), then these receipts can either do damage (by eventually increasing the denominator of the r_1 component) or do good (by increasing the numerator of both the r_1 and r_2 components). The specific gossiping algorithm that we propose stipulates that peers must show receipts of this type.

5.2.5 Analysis

A symmetry exists with regards to the computation of the r_1 component: When Peer P computes the numerator—the result of maxflow(P→C)—Peer P has been informed about receipts of the form $X \rightarrow C$ directly from the visiting peer, who brought them to his attention through the gossiping algorithm. Therefore, there is additional information for the area of the graph around vertex C and the edges that point *to* it. At the same time, Peer P only knows about the area around vertex P and the edges that point *away from* it indirectly, through the gossiping algorithm. In an analogous way, when Peer P computes the denominator of the r_1 component—the result of maxflow(C→P)—he knows directly about receipts of the form $X \rightarrow P$, because these are the receipts that visiting consumers signed and gave directly to Peer P after their transactions with Peer P. Although the prospective consumer tries to hide them, Peer P learns more about the graph area around vertex C and the edges that point *away from* it, indirectly, through the gossiping algorithm. Therefore, the computation of the r_1 component proceeds in an unbiased way: Peer P has a lot of information for the edges pointing *to* the target vertices for both maxflow computations, but less information for edges that point *away from* the source vertices. Therefore, the ratio of the two maxflow results is not influenced because the loss of information in both the numerator and the denominator is caused by the same underlying process—the gossiping algorithm.

One difficulty with gossiping is that because members may present many receipts *before* requesting service, team servers may not have time to verify the signatures on all of them in real time (for signature verification times see Chapter 7). Team servers could verify

receipts in a background process. However, if team servers allow unverified receipts to be stored and used in maxflow computations, they would have to verify them if these receipts end up forming part of one of the maxflow-discovered paths.

Another problem with the gossiping algorithm is that it makes the peers susceptible to DoS attacks at the "application" layer. Malicious consumers could overload peers with receipts that are fake. A malicious consumer has nothing to gain from this type of attack. One possible defense would be not to merge receipts if the consumer turned out to be a free-rider based on the reciprocity algorithm. In any case, this attack cannot hurt the system as a whole, only individual peers/teams (see also Chapter 8—Discussion and Future Work).

Note also that with its FIFO rule, the gossiping algorithm fulfils the requirement for keeping short-term history only. There are many possible variants to the receipt replacement discipline, but the FIFO version that we presented above is the one that we will evaluate in the next chapter. We believe that it achieves its purpose of realizing a useful receipt graph in a decentralized way.

5.3 Bootstrap algorithm

5.3.1 Introduction

The bootstrap algorithm is the third and final algorithm used in P2PWNC. Peers in P2PWNC need to contribute first before they start to consume. This is because the reciprocity algorithm searches for direct or indirect debt from a prospective contributor to a prospective consumer. If the consumer is new and has never contributed then he would be no different from a free-rider according to the Subjective Reputation Metric (SRM) computed by the contributor. However, if the new peer attempts to use the reciprocity algorithm in order to contribute service, he will, in turn, find that all other peers appear as free-riders to him: this time, the SRM will be zero because the new peer has no outgoing receipts—exactly because nobody has yet contributed service to him.

To break this deadlock, the bootstrap algorithm specifies that whenever a new Peer N joins a P2PWNC community, Peer N will start contributing at first without running the reciprocity algorithm. (Peer N will use, however, the gossiping algorithm proper, that is, Peer N will conduct merging of receipts when a consumer from N requests service.) At the same time, we assume that the members of Team/Peer N will be trying to consume in the community. At first, they will be unsuccessful, if they have no incoming receipts of the form $X \rightarrow N$ to show during the gossiping protocol, and if no receipts of the form $X \rightarrow N$ are

stored in the prospective contributor's receipt repository. However, as soon as the first receipt of the form $X \rightarrow N$ is earned, the probability of Peer N receiving service by peers who follow the reciprocity algorithm becomes positive. Peer X, in the example above, would need to be a good contributor in the P2PWNC community himself, so that others that owe directly or indirectly to X will now owe indirectly to N. Of course, Peer X now owes to N, so N could request service from X and potentially receive service. However, if X runs the reciprocity algorithm, whether and how much he cooperates will depend on the SRM of N that X computes.

During the bootstrap phase, Peer N is also exposed to attacks by free-riders. By not running the reciprocity algorithm, and by contributing service and accepting receipts from anyone, Peer N may inadvertently cooperate with free-riders.

5.3.2 Heuristic

A heuristic we propose for the bootstrap algorithm is the following: Each peer maintains a bootstrap parameter called *patience*. As the members of Peer N try their luck in the community, some of them will eventually receive service and issue a receipt. As soon as Peer N manages to issue a total of *patience* receipts, Peer N will stop the bootstrap phase, and start using the reciprocity algorithm properly. The idea is that, this way, Peer N may deduce that he has become "well-known" as a contributor in the P2PWNC community.

If the patience parameter is set to zero, we discussed above that the new peer will appear no different from a free-rider to other peers. If the patience parameter is set to only one, the new peer is overconfident: There are many reasons why a peer could have managed to consume service without in reality being a "well-known" contributor in the community. One reason is for Peer N himself to receive service from a peer that is also undergoing bootstrap. The higher the patience value is set, the more chances Peer N has of issuing receipts to peers that successfully recognized Peer N as a good contributor, which was the initial goal of N; otherwise, Peer N would have to return to a bootstrap phase.

The meaning of the bootstrap phase is that a peer is patient when first joining, paying his dues to the community in order to be recognized after that as a good contributor. This is an aspect of the social cost of the free-ID regime: If we let P2PWNC peers cooperate with newcomers we give incentives to free-riders to constantly switch to new IDs before requesting service. If we let P2PWNC peers be strict with newcomers (our choice), newcomers will have to prove themselves to the community first. A practical way to know

that they *have* proven themselves to the community is to succeed in a number of transactions in which they participated as a consumer.

We show in the next chapter that this heuristic is good enough for peers to use when first joining a P2PWNC system. Simulations show that the value of patience can be relatively small (we use 10).

Figure 5.6 shows the interplay of the three algorithms that we discussed.

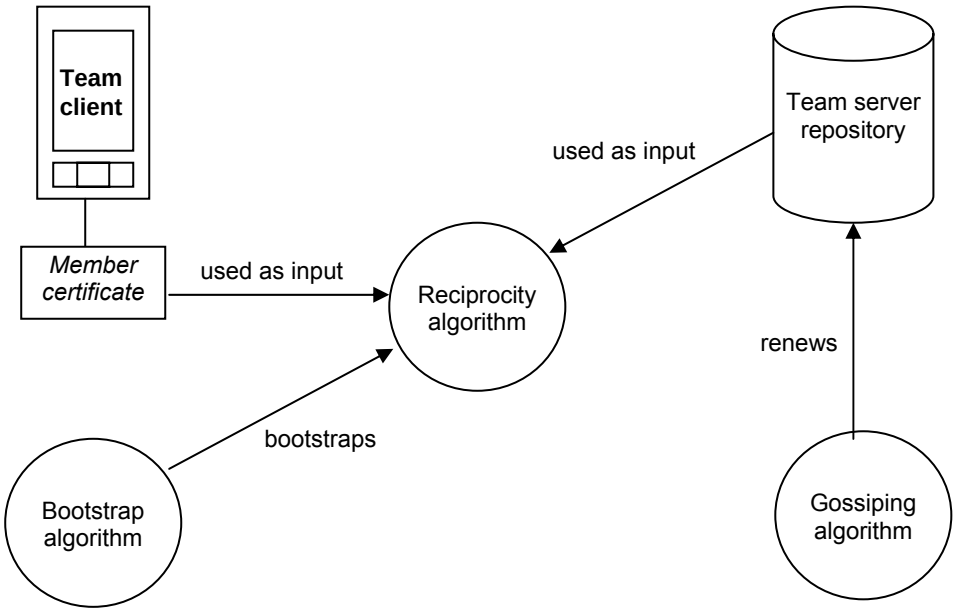

Figure 5.6 The interplay of the reciprocity, gossiping, and bootstrap algorithms. The bootstrap algorithm is used to let peers become "known" contributors in the community before they start to use the reciprocity algorithm. The gossiping algorithm renews the view of the receipt graph that each team has by renewing the team server's repository of receipts. The team's repository of receipts is used as input to the reciprocity algorithm, along with the certificate of the current visitor, to decide whether and how much service to contribute to this visitor.

Chapter 6

Evaluation

6.1 Evaluation framework

Our objective with the simulation-based experiments that follow was not to perform low-level WLAN-aware simulations of P2PWNC. The results in this chapter apply to other electronically mediated P2P communities that use a receipt-based incentive technique like the one we described (under certain additional assumptions, all of which are described in the introductory section below). Our main objective was to see if a completely decentralized P2P system can attain good (in a manner to be defined) levels of cooperation among the peers that comprise it. In constructing an evaluation framework for P2PWNC, we borrowed ideas from [71], which we expanded using additional concepts from Cooperation Theory [57, 68, 69]. Below, we present the concepts of the evaluation framework that we use, and the specifics of our custom P2PWNC simulator. Definitions for key terms that are used throughout the chapter appear in bold.

6.1.1 Modeling teams, receipts, receipt repositories, and the gossiping algorithm

We have already mentioned that the peering entities in P2PWNC are *teams*. Although teams serve a practical purpose, the concept of teams will only appear briefly in the experiments that follow. Most of the experiments involve interacting *peers* (which are used to model entire P2PWNC teams) that contribute and consume service from and to each other. Note here that we will be using the male pronoun “he” to describe a peer/team in the analysis that follows and in many of the experiments.

The first simplification we made to the P2PWNC architecture for the purposes of evaluating it in this chapter was the adoption of *unit-weight receipts*. All receipts generated during our simulations are assumed to have the same weight, which for simplicity we set to one. In the section that follows, we will discuss how this unit-weight model can be reconciled with real-world use of a P2PWNC system. Unit-weight receipts do not affect the maxflow-based algorithms that we use. The definition of the logical receipt graph from Section 5.1 still applies, with the additional understanding that the receipts that realize the graph have weight equal to one.

We also simplify the P2PWNC identity system. The fact that peers use public keys as their public IDs in P2PWNC is not important. Internally, our simulator keeps track of different peers using an integer counter-based scheme, and we do not model the signature generations or verifications that are involved in the receipt generation protocol.

We *are* interested, however, in both types of repositories that peers maintain, that is, the receipt repository found on each team server, and the mobile repositories found on P2PWNC clients. From now on, we will simply refer to them as the **server repository** and the **client repository** respectively. Their sizes, which could be different, are an important simulation parameter. Size is measured in receipts. Measuring the size in receipts is equivalent to measuring it in bytes: Receipts in P2PWNC are constant in size and this size depends only on the specific public key algorithm (for example, RSA-1024 or ECC-160) that the P2PWNC community uses (see also Chapter 7—Protocol and Implementation).

When we refer to "the peers" in our experiments, we are focusing on P2PWNC teams that consume and contribute without modeling specific team members. We make another simplifying assumption that concerns peers and their receipt repositories: We said above that each peer has a server repository and a client repository. The assumption now is that, at any time, the client repository contains the exact same receipts as the server repository; that is, we do not model the *update* phase of the gossiping algorithm (see Section 5.2). Of course, if the client repository is smaller than the server repository, then, according to the gossiping algorithm, the client repository contains only the newest receipts from the server repository. Receipt timestamps are implemented in our simulator using an integer counter-based scheme; that is, receipts are numbered according to the order of their creation, and this number serves as a unique receipt ID. (In a real implementation, the unique receipt ID is the tuple of receipt fields containing (1) the consuming member certificate, (2) the contributing team's public key, and (3) the receipt timestamp.)

In addition, a peer's client repository never contains receipts that show the consumption actions of this peer. We already discussed that a peer has no incentive to show these receipts to prospective contributors and a peer's team server never includes these receipts when answering to the update requests of its members (see Section 5.2).

We are also interested in the incentives of members that may wish to hide the receipts they sign from their team server (see Section 4.7). Therefore, as per the P2PWNC algorithms, we do assume that a peer in our simulations does not store his own outgoing receipts in his server repository unless he discovered them through the gossiping algorithm

in the manner we described in Section 5.2. Note, therefore, that even though we do use peers to model entire teams, we do not assume that these peers have complete knowledge of their consumption history. We do all these because we are interested in modeling all specific incentives-related aspects of the gossiping algorithm.

Finally, the merging phase of the gossiping algorithm is implemented in the simulator as specified in Section 5.2.

6.1.2 Modeling mobility, community growth, and the bootstrap algorithm: The concept of "rounds"

Our simulation runs consist of **rounds**. The communities that we simulate contain at any one time a number of peers, say *N*. A round is set of *N* **matches**. A match is a prospective P2PWNC transaction in which two peers, the consumer and the contributor, are paired. We assume this happens because the consumer visited (the area near) the contributor while following the movements dictated by one of our **mobility models** (see more below). A **visit** in our model always implies a request for P2PWNC consumption. It is then up to the prospective contributor to decide whether or not he will contribute to the prospective consumer. If he does, we say that the transaction was successful and that the contributor **cooperated** with the consumer.

We have defined three mobility models, which we call **"perfect matching," "preferential visitations,"** and **"random waypoint."** We will present the latter two in the sections where the related experiments appear. We define "perfect matching" below.

In "perfect matching" each round consists of *N* matches where *N* is the current number of peers in the community. The visits that resulted in these matches are generated randomly but they also follow a rule: In each round, every peer in the community will take part in two matches, in one as prospective contributor and in one as prospective consumer. That is, in each round of "perfect matching" a peer gets exactly one chance to contribute and exactly one chance to consume.

This simplified mobility model is influenced by similar matching models in evolutionary games [57, 68, 71, 81, 82] and is only a rough approximation of how teams would interact in a real P2PWNC system. The other two mobility models we propose expand on "perfect matching." The distinctive characteristic of "perfect matching" is that it effectively assumes that all peers have the same consumption request rate.

We also simulate **community growth**. In the simplest case, we assume that as the simulation rounds progress, one new peer enters the community every round, up to a specified maximum community size, which is a parameter of the experiment. We further assume that peers never leave a P2PWNC community. This assumption is influenced by the nature of a P2PWNC system: Unlike a file-sharing system in which peers use file-sharing software on PCs, a P2PWNC contributor uses a WLAN AP, an embedded device that will probably stay powered-on continuously. We therefore assume that a P2PWNC peer, after he joins the community during the community's growth stage, will stay in the community forever. Of course, as we will see in the experiments below, being part of the community does not necessarily mean that a peer is cooperative: Free-riders may also be part of the community (see the various *strategies* below, in Section 6.1.4).

We also implement the bootstrap algorithm according to its specification in Section 5.3. A P2PWNC peer that first joins the community is *patient*, that is, he awaits for a number of *successful consumptions* (see Section 6.1.3 for a definition) before he starts using the reciprocity algorithm. Before that, we assume that he cooperates with anyone requesting service. When the peer eventually issues a number of receipts equal to the *patience* parameter, this means that the peer successfully consumed *patience* times and he switches to using the reciprocity algorithm.

6.1.3 Modeling benefit and cost

The benefits and costs of participating in P2PWNC are difficult to quantify. There are many indirect costs and benefits involved. As an example, a direct cost of participating in P2PWNC is the extra volume-based charges that a contributor would need to pay to his ISP because of the extra traffic he relays for P2PWNC visitors. However, many Internet subscriptions use a flat-rate scheme. Below we present a list of potential cost generators that we have identified (some of them have been presented before in Section 4.4, where we discussed the issue of congestion in P2PWNC), starting from the ones we consider less important, continuing to the ones that we consider more important.

Cost generators

First, there is the issue of radio power. In order to contribute to a visitor, a WLAN AP will have to transmit and receive packets wirelessly; in order for the packets to reach a visitor who is at the edge of an AP's coverage radius, the AP's transmitter may have to

reach its maximum allowable power limit (which in many European countries is set to 100 mW [12]). If the AP is inside a household, the effect of this radiation will be greater inside the residence than outside. Therefore, we can identify an indirect cost in the form of a non-zero probability for adverse health effects for the members of the household.

A second cost generator is the problem of security attacks against an owner's intranet, assuming one exists. Even though we can expect P2PWNC implementations to be reasonably secure, by allowing the possibility of foreigners to access the owner's AP and backhaul connection, an AP owner potentially exposes his private network to security attacks.

A third cost generator (mentioned previously) is potentially more direct: If the owner's Internet connection is metered, or if it is on a flat-rate subscription but with a maximum allowable traffic limit beyond which it becomes metered, every additional Internet packet the owner relays for visitors is a potential cost generator.

A fourth cost generator is reduced service level for the owner when a visitor is accessing the owner's Internet connection. Although we have assumed that congestion is not an issue in P2PWNC (see Section 4.4), we expect that P2PWNC implementations will allow owners to allocate an amount of bandwidth to be used by P2PWNC visitors. When visitors use this bandwidth, it is not available to the owner any more. Thus, even with a technically flawless quality-of-service system, there is always opportunity cost to the owner because he could have used this bandwidth if it was available, and he is not able to use it if he allows visitors to relay packets over his connection.

A fifth cost generator is the issue of violating an ISP's Acceptable Use Policies. If the ISP does not allow the sharing of a residential Internet connection with outsiders, then, every time a visitor is allowed to relay his packets, the owner of the connection acts in breach of contract. This is a potential cost generator: It increases the probability that the ISP terminates the contract or takes legal action.

Benefit generators

We will now examine the potential benefits of being a consumer in P2PWNC. First, consumers use a service that is equivalent to 2G/3G cellular service without paying the metered charges usually associated with these services. Second, even though we mentioned above that WLAN APs (and clients) can transmit up to 100 mW of power, this is still an order of magnitude less than the maximum power that 2G/3G cell phones can trans-

mit [14]. Therefore, there are potential health benefits for the P2PWNC consumer. Third, the maximum throughput than can be achieved through a combination of WLAN APs and DSL/Cable connections is an order of magnitude higher than the best 3G offerings [83].

Note that the benefit generators above were identified using 2G/3G cellular service as a baseline for comparison. This is a conservative approach. We could also examine the P2PWNC service isolated from the fact that (2G) cellular service is ubiquitous. In that respect, the benefit of using a cellular-like service (the P2PWNC service) is substantial if the willingness of current cellular subscribers to pay for cellular service is a good indicator of the benefit they receive from it.

Benefit function

Having identified several potential benefit and cost generators, we now proceed in presenting the specific benefit and cost values that we will assign to the P2PWNC service in our simulator model. These values will become important later on when we compare the various *strategies* that peers may follow. For a *shortsighted rational* peer (see Chapter 3), if a strategy confers more net benefit than his current strategy in the short term, this causes the peer to *switch* to that strategy with a non-zero probability (see also Section 6.1.6).

Starting with costs, the cost generators we identified above all confer cost that depends on the number of visitors allowed to access the owner's WLAN AP. We will assume the simplest possible cost function: In our simulations, each time a consumer is allowed to access the AP of a contributor, the contributor pays a cost of one unit. This one unit is our attempt to model all the previous cost generators; the distinctive characteristic of our cost function is not the actual value, but the fact that it is linear to the number of visitors allowed, or, equivalently, linear to the number of times a contributor cooperated with a consumer. An important simplification is that this cost is independent of the characteristics of the session of the consumer. This means that the cost for a contributor is the same irrespective of what WLAN/Internet application the consumer uses, and for what duration. Below, in order to have a concrete example of such an application, we will assume that the only application that P2PWNC consumers use is to place and receive calls of a fixed duration.

Equivalently then, we can define a maximum benefit, b_{max}. This is the maximum possible benefit that a consumer can receive every time he is allowed to access a foreign WLAN AP and use the telephony application. Because the cost to contributors, denoted c,

is one, b_{max} equals the maximum benefit to cost ratio of the P2PWNC service, an important input parameter for the experiments below.

According to the reciprocity algorithm, a *Subjective Reputation Metric* (SRM) characterizes each consumer from the perspective of the contributor. Here, we make the following modeling decision. We will use an *SRM-to-benefit* function that we assume is the same for all peers, and which is programmed in our simulator. According to this function, a consumer will receive an amount of benefit from each consumption that is a linear function of his SRM. This implies that contributors can directly control the amount of benefit the consumers receive, which is a modeling leap. However, it is reasonable to assume that the contributors have a way to reduce the service quality that they offer according to the SRM. We assume there is a universal function for translating SRM to service quality. Then, we assume that a universal function exists, which translates this amount of service quality to benefit for the consumer.

Let us think of a concrete example, assuming that the standard application is telephony with calls of a fixed duration. Then, a way to reduce the benefit of consumers while keeping the cost of the contributor approximately the same is to *delay* their login to the AP for an arbitrary amount of time. For lower SRMs, this delay can increase exponentially for example. We can thus define an *SRM-to-benefit* function in the following way: For values of SRM below a threshold, SRM_{th}, prospective contributors do not cooperate with the contributor. We assume a universal SRM_{th} that equals 0.1 in our simulations; a consumer with an SRM below this value is considered a free-rider from the perspective of the contributor. For values of SRM above this threshold, the contributor will cooperate with the consumer. The benefit the consumer will obtain will be linear to his SRM value. For SRMs that equal one, the consumer will obtain a benefit of b_{max}. Figure 6.1 shows the exact shape of the SRM-to-benefit function.

If we assume that all P2PWNC consumers want to make or receive a two-minute video call, we can see that the costs to contributors will be approximately the same per video call. Then, one way for P2PWNC contributors to punish prospective consumers is to delay their login to the AP exponentially, depending on their SRM. We assume, however, that the consumers will issue a unit-weight receipt no matter how low the benefit they receive is, because we assume consumers *want to complete* the call. The receipt generation protocol (see Section 4.6.2) will not allow them to complete it if they fail to sign receipts.

In closing, from now on, when we refer to the **amount of cooperation** or the **cooperation level** we will refer to the benefit that a consumer received from a contributor.

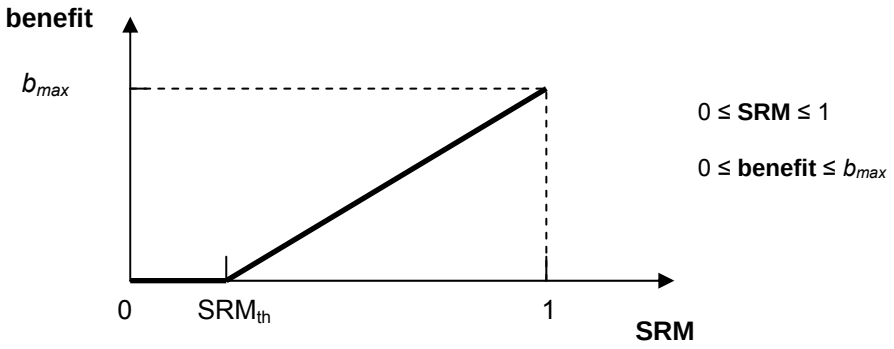

Figure 6.1 The shape of the SRM-to-benefit function used in the experiments of this chapter. After the contributor computes the SRM of the consumer, he may delay the login of the consumer in order to lower the consumer's benefit. The functions that translate SRM to delay and delay to benefit are not required—we simply assume that the SRM-to-benefit function is as above.

When we say that a contributor **cooperated fully** or **cooperated to the maximum**, we mean that the consumer received a benefit of b_{max}. In addition, we define a **successful consumption** to be any consumption request that resulted in some amount of cooperation (that is, the SRM for the consumer in the eyes of the contributor was above the threshold), and which resulted in the issuing of a unit-weight receipt by the consumer.

6.1.4 Strategies and strategy mixtures

From now on, we will use the term **strategy** to describe the behavior of a peer in relation to his contribution decisions in the matches in which he is the prospective contributor. For example, peers that follow the reciprocity algorithm that we specified are said to follow the **RECI** strategy (short for RECIprocity). Equivalently, we will call these peers the **followers** of RECI (or RECI's followers). In several of the experiments, the objective is to compare the RECI strategy to other strategies (in a manner to be defined below—see Section 6.1.5).

We have defined three additional strategies called **ALLC**, **ALLD**, and **RAND**. ALLD and ALLC correspond to a *free-riding* and an *altruistic* strategy respectively (see Chapter 3 for definitions). An ALLD follower is a P2PWNC peer that never cooperates with anyone requesting for service. At the same time, an ALLD follower may request

service from other peers. An ALLD follower does not run the reciprocity algorithm; therefore, he also does not run the gossiping algorithm when acting as a consumer because he has no receipts in his client repository to show—his server repository is always empty. (Because ALLD followers deny all service requests, they also do not get the chance to copy receipts from prospective consumers to their server repositories through the merging phase of the gossiping algorithm.) In addition, because they never cooperate, they never earn any receipts pointing directly at them. The ALLD strategy models P2PWNC participants that have reprogrammed their WLAN APs to disallow visitor logins, or have turned off their WLAN APs completely, in an attempt to minimize their costs.

On the other hand, the ALLC strategy corresponds to a P2PWNC participant that has decided not to use the reciprocity algorithm but to cooperate with anyone requesting service. ALLC followers, however, follow the gossiping algorithm, and, when acting as consumers, they show the receipts from their client repository in order for these receipts to be merged with the contributor's server repository.

RAND is the third strategy that we defined. A RAND follower, according to an internal probability p_c that may be different for each RAND follower, decides in every match (where he is the prospective contributor) whether to act like ALLD or ALLC. When acting as ALLD, he does not cooperate and he does not accept receipts from the consumer for merging. When a RAND follower decides to act as ALLC, he cooperates with anyone that requests service. In that situation, he will also merge the receipts given to him by the consumer with his server repository. We assume these receipts are copied to the client repository of the peer in the update phase of the gossiping algorithm.

The RAND strategy models a P2PWNC peer that does not rely on the community's history to make his contribution decisions. In the "perfect matching" mobility model a RAND follower is an *under-contributor*: that is, although he gets one chance to consume in every round, he may not contribute in that round, thus departing from the 1-to-1 contribution-consumption ratio that we attempt to enforce with the reciprocity algorithm. In addition, a RAND follower adapts its probability of cooperation, p_c, according to an internal process. He does all these in an attempt to maximize his net benefit. More specifically, the RAND strategy adapts the value of p_c. Every RAND follower checks his score every 50 rounds and compares it with the previous 50 rounds. In the beginning, p_c equals 0.5. After 50 rounds, a RAND follower will increase this probability by 0.1. If in the current window of 50 rounds the peer did better or the same (in terms of an increase in

score) as the previous, he continues increasing this probability by 0.1 every 50 rounds, otherwise, he decreases it by 0.1, and continues decreasing it by 0.1 every 50 rounds if his score continues to increase or stay the same compared to the previous 50 rounds. If not, he changes direction again. Of course, the value of p_c cannot exceed one or be lower than zero.

When followers of these four strategies are present in a P2PWNC community we will also say, equivalently, that the community is a **strategy mixture** of the strategies that are represented in it. In addition, for simplicity, instead of referring to ALLC followers, ALLD followers, RECI followers, and so on, we will simply refer to ALLCs, ALLDs, and RECIs. A strategy mixture may be specified in absolute numbers (for example, a community of 100 peers, of which 25 are ALLCs, 25 are ALLDs, 25 are RANDs, and 25 are RECIs), or in percentages (a community with 50% ALLDs and 50% ALLCs).

A special case, which we label "**strategy mixture (probabilities)"** in our simulation input parameters, is the following. When we say, for example, that the "strategy mixture (probabilities) is 33% ALLC, 33% ALLD, and 34% RAND" this means that, as the P2PWNC community grows to reach its maximum number of participants, the new peers that join it will (initially) follow one of the strategies specified in the mixture according to the probabilities specified above.

A final strategy that we have defined is **pre-RECI**. This represents the altruistic/ bootstrap phase of the RECI strategy. When a RECI follower joins the community (or when a peer switches to the RECI strategy during the simulation because of *evolution*—see Section 6.1.6) he will follow the pre-RECI strategy until he successfully consumes for a number of times specified by the *patience* parameter of the bootstrap algorithm.

Finally, we must specify the amount of benefit that consumers receive from pre-RECIs, ALLCs, and RANDs (when RAND decides to act as ALLC). We have specified the *SRM-to-benefit* function that the RECI strategy uses to contribute benefit in the previous section. Here, we simply require that when pre-RECI, RAND, and ALLC decide to cooperate, the amount of benefit received by the consumer in that transaction is b_{max}, that is, the maximum amount of benefit possible.

6.1.5 Scores, ratings, and social welfare

Here we will define the metrics that play a central role in the experiments that follow. A **peer's score** is the sum of the benefits he has received minus his costs (in that respect, it is the same as his *net benefit*—see Chapter 3—however, we will use the more

intuitive term *score* in this chapter). A peer's score may increase as rounds progress, it may stay the same, or it may decrease. A score can also be negative if the costs exceed the benefits. A peer's score is set to zero whenever he changes *strategy* (see also *evolution* in Section 6.1.6 below).

A **peer's rating** is a peer's score divided by the number of rounds it took to attain it. In that respect, the rating is the (running) average score per round.

A **strategy's rating** is a weighted average of the ratings of its followers; the weights are calculated according to how many rounds each follower has been using the strategy. Equivalently, if we note that a peer's rating is a peer's score divided by the number of rounds it took to attain it, a strategy's rating is the sum of the scores of its followers divided by the sum of the rounds it took each peer to attain his score. In a mixture composed only of RECI, and under the "perfect matching" mobility model, the **optimal rating** of RECI is $b_{max} - 1$. This is the rating that RECI would attain if all its followers contributed b_{max} benefit to all consumers (also RECI followers) whenever they requested service, at a cost of $c = 1$ to themselves. In such a case, in every round of "perfect matching" a peer's score would increase by $b_{max} - 1$, which would also equal the peer's rating, and because all peer ratings would be the same in such a case, it would also equal the strategy's rating.

Social Welfare (denoted **SW**) is the sum of the scores of a community's peers. The **optimal SW** is the SW that would be generated in a community if, in all matches and in all rounds, the prospective contributor contributed b_{max} benefit to the prospective consumer, at a cost of $c = 1$ to himself. That way, SW would increase by $b_{max} - 1$ after every match.

The **fraction of optimal SW** at a specific simulation round is the ratio of the SW until that round divided by the optimal SW until that round.

6.1.6 Evolution

In some of the experiments that follow, we have used an evolutionary framework in order to simulate P2PWNC communities. In short, we let the peers that are part of the community engage in an evolutionary game, in which certain strategies may prevail. Evolutionary games are used to model adaptive agents with shortsighted rationality [69] (see Chapter 3 for a definition). When we say that we allow our peers to **evolve,** we mean that we allow them to **switch** to another strategy during the course of the simulation. Doing

so, allows them to explore the strategy space in search for strategies that achieve higher ratings.

To model the precise way with which evolution takes place we attempted to predict when, why, and how the peers in a real P2PWNC community would switch to different strategies: If a system like P2PWNC were operational, we assume that the peers who participate in it would use online forums to discuss aspects of the system. In these online discussions, peers would share their experience and give advice to other peers on several P2PWNC issues, such as which protocol implementation to use. In addition, these online forums would be the place to find and download tampered versions of the P2PWNC algorithms that would promise to maximize the net benefit of the peers that use them.

We also assume that followers of a particular strategy (equivalently, users of particular software) would post their thoughts on how their strategy performs. For example, if a P2PWNC community contained many pre-RECIs and ALLCs, then it is conceivable that a follower of the ALLD strategy would be content with the cooperation levels he experiences as a consumer. He would then suggest to other peers to turn off their P2PWNC APs to minimize their cost because this is what he did and saw no reduction in the benefit he received while consuming.

We model this online forum environment in the following manner. We first define the **probability to mutate**, $p_{m.}$ During a simulation run, at the end of a round, a peer decides to switch randomly to any strategy from the set of available strategies (see Section 6.1.4). When a peer switches to another strategy, he sets his current score to zero, and he sets a variable indicating *when* he switched to this strategy. Note that a peer's rating carries specific weight when computing the rating of a strategy, a weight which depends on how long the peer has been using this strategy. In our online forum scenario, this models the difference between a peer who has recently started using a new version of his software and is still reluctant to publicly express opinion on the effectiveness of this new software, and between the "veterans" of that software/strategy.

With mutation, we allow strategies that are not present in the current mixture of strategies to actually appear. If these strategies did not appear, nobody would know whether they would be effective or not. In that sense, mutation is a generating force. For practical reasons, out strategy set is limited (to ALLC, ALLD, RAND, and RECI) but we believe that these four strategies (RECI, and three probabilistic strategies) are the probable strategies that would coexist in first-generation P2PWNC systems. More sophisticated

strategies that try selfishly to obtain more net benefit from the community would appear later; we will actually experiment with such a strategy, which exploits the vulnerability discussed in Section 5.1.7.

In addition, mutation is important in that it can make strategies that were present and disappeared from the mixture to return, perhaps under conditions that are more favorable for them. We note that in evolutionary settings the success of a strategy depends on the specific mixture of strategies and on the available strategies that can appear through mutation. We return to this subject below (when we discuss the subject of *invasion*).

Second, we define the **probability to learn**, p_l. During a simulation run, at the end of a round, a peer may *learn*. That is, a peer may choose another strategy from the ones *currently available in the mixture*. This is not enough to complete the switch though. The peer will then compare the two strategies, his current and the new one, according to outside input (for example, in the form of online forum postings). The important detail here is that the peer will not just rely on his own rating, but he will attempt to approximate the rating of his current strategy and the proposed strategy. In order to model this procedure, which is highly susceptible to noise and misinformation, we say that a peer will switch to that strategy with a *probability* that is *proportional to the difference in rating* of the two strategies. That is, after a peer using strategy OLD decides to learn (with probability p_l) and gets ready to switch to a another strategy from the ones available currently in the mixture, NEW, he will only actually do it with probability:

$$p = 1 - \frac{rating_{OLD} + 1}{rating_{NEW} + 1} \tag{6.1}$$

(We add one unit to both the numerator and the denominator because, according to our definition, a strategy's rating can be negative, with a minimum of –1 in the perfect matching mobility model. In the other two mobility models because a contributor may contribute more than once during a round it is conceivable that his rating will be even lower than –1, possibly causing his strategy to have a rating lower than –1. In the experiments with evolution, we only use the perfect matching mobility model though.)

In closing, we describe how the pre-RECI strategy fits inside our evolutionary framework. The pre-RECI phase is a temporary phase that RECI followers go through, knowingly. Although pre-RECI is no different than ALLC, pre-RECI followers do not

learn, though they may mutate. That is, they are not "envious" of other strategies in the manner we described above. In that respect, we say that pre-RECIs at the time act with forward-looking rationality: They are making a conscious choice to accept a potentially lower rating temporarily, in order to switch to RECI, which was their initial choice. For the same reason, any peer that decides to switch to RECI either through mutation or through learning first goes through the pre-RECI phase.

6.1.7 Simulator inputs and outputs

The *input parameters* that we use throughout the experiments are:

1) **The number of peers** This is the maximum number of peers that will join the community.
2) **Community growth rate** This is the rate with which new peers join the community, usually measured in peers per round. In certain experiments, the simulation starts when all the peers already have joined the community.
3) **Server repository size** This is the server repository size, measured in receipts, and it is the same for all peers.
4) **Client repository size** This is the client repository size, measured in receipts, and it is the same for all peers.
5) **Patience** The value of the patience parameter of the bootstrap algorithm, and it is the same for all peers.
6) **Benefit function** We vary the shape of the universal SRM-to-benefit function (see above) by changing $\boldsymbol{b_{max}}$.
7) **Mobility model** "Perfect matching" (defined above), "preferential visitations" or "random waypoint" (see experiments).
8) **Strategy mixture** The strategy mixture, or the initial probabilities with which peers who join the community will follow one of the strategies stated in the mixture.
9) **Evolution** The values for the probability to mutate and the probability to learn. For $p_m = 0$ and $p_l = 0$, no evolution occurs.

The main simulation *output parameters* of interest are:

1) **RECI's rating**, when it is alone in the mixture, as rounds progress.
2) The changing number of a **strategy's followers** as rounds progress.

3) The **fraction of optimal SW** attained as rounds progress.

4) A **peer's score** as rounds progress.

5) **SW** as rounds progress.

6.1.8 Guide to the experiments that follow

There are two groups of experiments that follow, Group A and Group B. In Group A, the strategy mixture is always composed of RECI followers alone, with two exceptions when a sophisticated adversarial strategy is introduced. In Group B, we involve also the ALLC, ALLD, and RAND strategies. In Group B, certain experiments include *evolution*.

6.2 Experiments: Group A

6.2.1 Experiment A-1: Cooperation level and information

Table 6.1 Simulation parameters for Experiment A-1

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	10, 500, 1000—see experiment
Client repository size (receipts)	10, 100, 200—see experiment
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	100% RECI followers
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

In this experiment, we wanted to show how the amount of information in the system affects the cooperation level attained. To do this, we experiment with various combinations of client and server repository sizes. These combinations are labeled in Figures 6.2 and 6.3 as "c: 10 s: 10," "c: 10 s: 1000," and so on, with the two numbers representing, respectively, the sizes of the client and server repository (in receipts).

For comparison purposes, we also experiment with an idealized P2PWNC system, where there is a central server with infinite capacity storing all receipts. We label this case "infinite central" in Figures 6.2 and 6.3. The central server acts as an honest super team server, in which all teams store the receipts that they earn. The server is honest in that it answers all queries by P2PWNC APs truthfully, running the reciprocity algorithm exactly like the AP's team server would run it, with the added benefit of having access to the complete receipt graph. Therefore, the central server returns to any querying AP the value of the *Subjective Reputation Metric* (SRM) for the prospective consuming peer that the AP wants to judge. Note that in the "infinite central" case, gossiping is not required and so the values for the client and server repository size do not affect the outcome of the simulation.

In this experiment (and in all experiments belonging to Group A), we do not allow for mutation or learning; therefore, since the community starts with 100% RECI followers it continues like that until the end of each simulation.

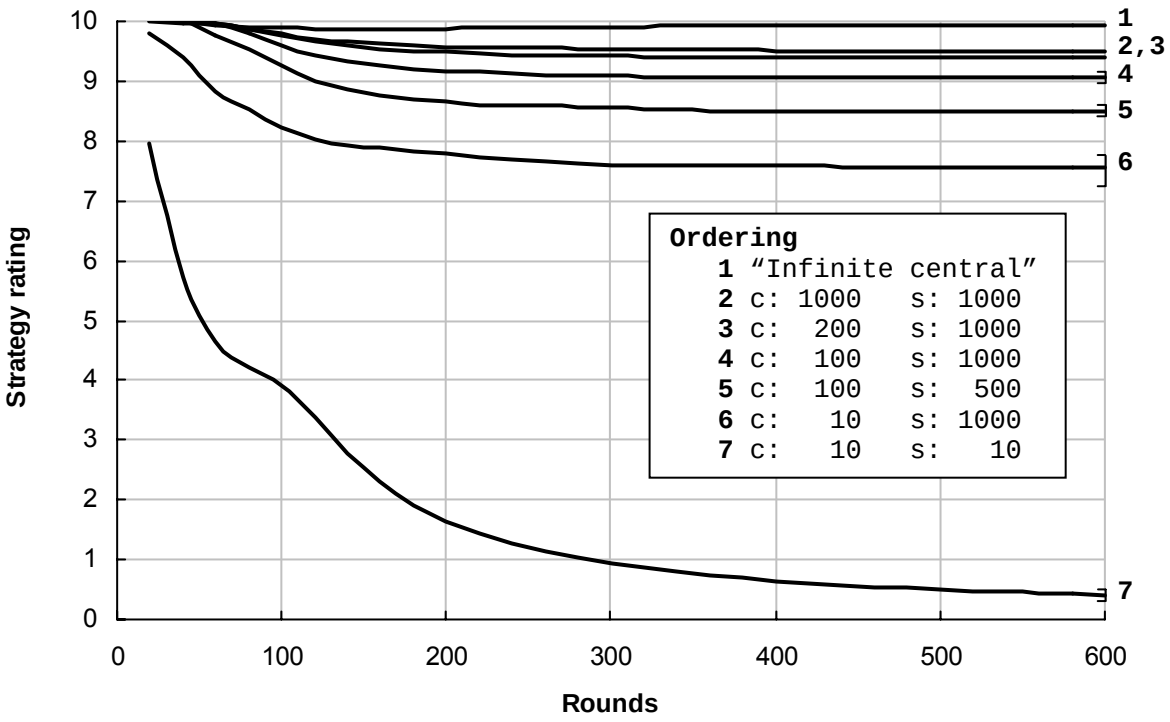

Figure 6.2 RECI rating (in a community of 100 RECI followers) as rounds progress, corresponding to seven different scenarios regarding the amount of information that is available in the system (see also Table 6.1). The legend shows the client and the server repository sizes, respectively.

In Figure 6.2, we plot the rating of the RECI strategy as rounds progress. We start the plot at Round 20 because the ratings vary greatly in the early rounds when the community is still small. In addition, at the end of each line, we plot a range corresponding to the 95% confidence interval obtained by our set of simulations. Note that each line in Figure 6.2 corresponds to one example simulation run, so its ending point need not coincide with the average found in the middle of the confidence interval. Note also that for the first three lines the confidence intervals are too small to see in this figure.

The first observation we can make by looking at Figure 6.2 is that the average rating of the RECI strategy practically stabilizes by Round 600 for the simulations represented by the top five lines. According to the b_{max} value of 11 that we use in this experiment, the optimal RECI rating is 10. In the "infinite central" case, RECI reaches an average rating of 9.94 by Round 600. We can see that there is still some efficiency loss compared to the optimal represented by a rating of 10, even in this special case. This results from how the reciprocity algorithm computes the Subjective Reputation Metric by relying on maxflow ratios that are rarely exactly equal to one, and by having a bias against ratios that are higher than one; and then, by the use of the specific SRM-to-benefit mapping

function that we adopted (see Section 6.1.3). In fact, the only situation in which we could obtain a RECI rating equal to the optimal rating is when we have infinite memory (in a scenario with a central server) and when there are only 2 peers in the community that take turns in consuming and contributing service from and to each other. On the other hand, this loss in efficiency is very small.

A second observation is that the lines follow an intuitive ordering. Line 2 depicts at Round 600 a RECI rating equal to 9.50, achieved when both the server and the client repository size is 1000 receipts. Line 3 depicts at Round 600 a rating of 9.40, achieved with the same server repository size (1000 receipts) and a client repository size of 200 receipts. Line 4 follows, with a rating of 9.04 at Round 600, with a client repository size of 100 receipts.

Note that increasing the client repository size brings only diminishing returns. For example, increasing it by 100 receipts (from 100 to 200 receipts) increases RECI's rating from 9.04 to 9.40. Then, increasing it by 800 receipts (from to 200 to 1000 receipts) increases the rating further, but only to 9.50. A second point is that the server repository size is important. By keeping the client repository size constant at 100 receipts and dropping the server repository size from 1000 receipts to 500 receipts (Line 5), RECI's rating drops from 9.04 to 8.50.

In addition, we see that even very small client repositories permit a good level of cooperation: for client repositories with only 10 receipts, RECI rating is 7.55 at Round 600.

Finally, the seventh (bottom) line shows what happens when the server repository is not large enough to support the number of peers in the community: RECI average rating is 0.41 at Round 600 (and continues to drop further). We can see in Figure 6.3, which plots the social welfare on the y-axis for the same set of simulations, that by Round 200, social welfare has practically stopped increasing (it increases, only very slowly). Most transactions result in small amounts of cooperation or no cooperation at all. This "cooperation collapse" happens because there is not enough information in the server repository (even after the merge with the client repository) to allow one RECI follower to identify another RECI follower by examining the community history. Most SRMs computed will equal zero, and therefore most prospective consumers will be mistaken for free-riders. As a result, the prospective contributor will not cooperate, no transaction will occur, and no new receipt will be generated.

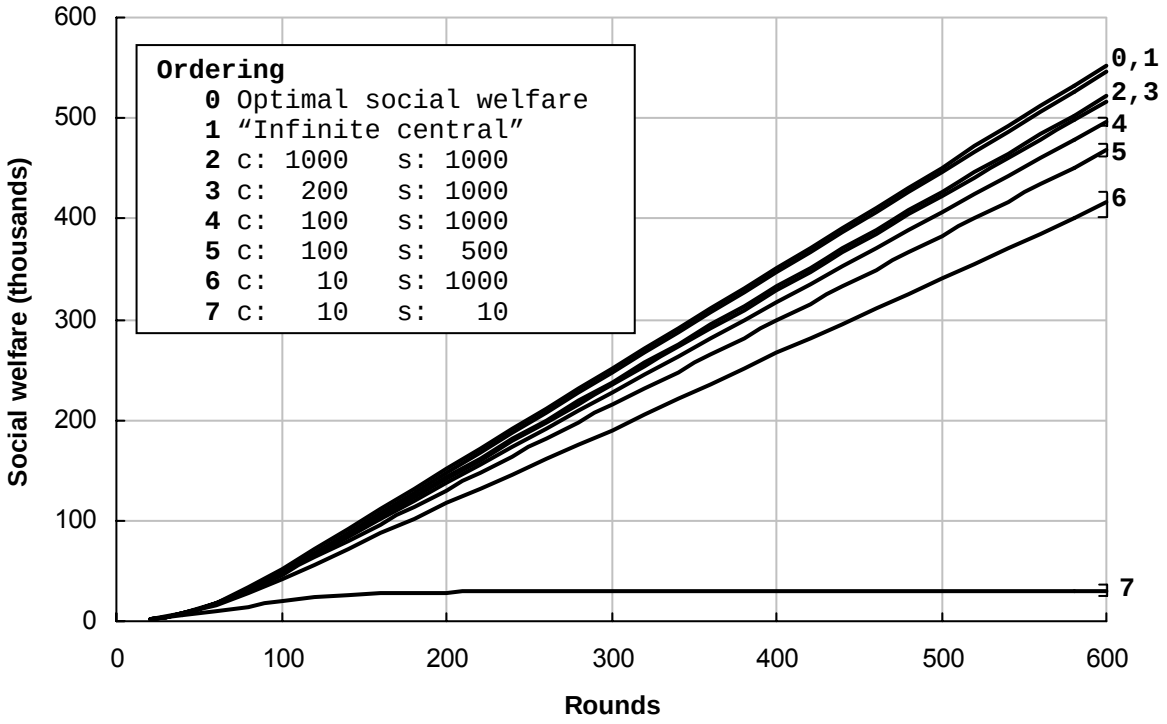

Figure 6.3 The increase in social welfare as rounds progress in the community of Experiment A-1, corresponding to the ratings shown in Figure 6.2. The legend shows the client and the server repository sizes, respectively.

In this example, with 100 peers and a server repository containing only 10 receipts, the chances for a non-zero result in a random maxflow(P→C) computation are small. Note that Figure 6.3 also plots the optimal SW as rounds progress for comparison purposes. The optimal SW corresponds to the constant RECI rating of 10 that would have been obtained had *all* matches resulted in full cooperation.

Note also that the drop in the seventh (bottom) line in Figure 6.2 increases its rate after Round 100. This is because until Round 100 there was always at least one new altruistic pre-RECI joining the community, that is, a RECI follower undergoing bootstrap. This fact kept average cooperation levels relatively high because pre-RECIs always fully cooperate. After Round 100, the remaining pre-RECIs slowly become RECIs when they finally achieve the 10 successful consumptions required by the patience parameter in this experiment.

In summary, we can see from this experiment that the decentralized storage sub-system based on gossiping can assist in attaining levels of cooperation that are close to an idealized centralized mode of operation.

6.2.2 Experiment A-2: The effect of patience

Table 6.2 Simulation parameters for Experiment A-2

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	100
Patience (consumptions)	0, 1, 2, 3, 10, 20—see experiment
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	100% RECI followers
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

In this experiment, we wanted to see whether the value of the patience parameter is crucial for the rating that RECI attains. In Figure 6.4, we plot six lines showing, from bottom to top, how the rating of RECI changes as rounds progress for patience values that correspond to 0, 1, 2, 3, 10, and 20 consumptions respectively. With the exception of the corner case of zero patience, all the other lines approach the value of 9.04 at Round 600, which we saw from the previous experiment is the value of the RECI rating that corresponds to the combination of client and server repository sizes that we chose for this experiment (100 and 1000 receipts, respectively).

A patience of zero means that as peers join the community they immediately start to follow the RECI strategy, without going through an altruistic pre-RECI phase in order to gather receipts. This way, no peer will ever cooperate with another peer. As a result, no receipts are issued and all SRM computations will result in zero SRM.

The ordering of the remaining lines (from bottom to top), as can be seen in the first 200 rounds, follows the ordering of the patience values, with a patience of 1 having the lowest rating during the first 200 rounds and a patience of 20 having the highest rating during the same rounds. This happens because, for higher values of the patience parameter, peers stay altruistic for more rounds on average compared to the number of rounds they stay altruistic if the patience value is lower. This means that at any specific round before Round 100 (when not all the peers have joined yet) there is, on average, a number of

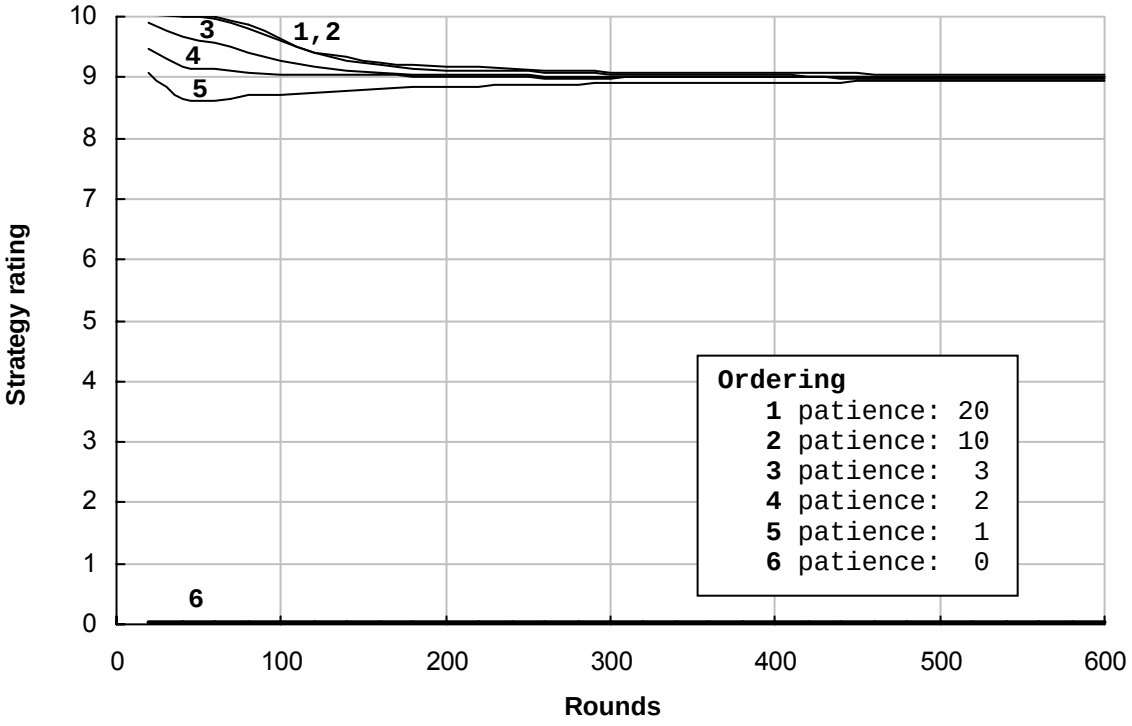

Figure 6.4 The effect of patience. For various values for the *patience* parameter of the bootstrap algorithm, the RECI strategy attains a similar rating as rounds progress (except for a patience of zero).

altruistic peers equal to the patience value, all of which are willing to cooperate to the full. This number is equal to the patience value because peers have only one chance to consume per round in this experiment because of the "perfect matching" mobility model. Therefore, when, for example, by Round 10, 10 peers have joined the community, the 10th peer will need at least 10 more rounds to switch from pre-RECI to RECI, the 9th will require at least 9 more rounds, and so on. In total, at Round 10, there will be 10 pre-RECIs. As one of them (the first one) switches to RECI in the next round (assuming that his consumption request will be successful), a new peer (in pre-RECI mode) will also join the community, keeping the total number of pre-RECIs equal to the patience value. At Round 100, the last peer will join the community (which will still have ten pre-RECIs total). After that round, it will take Peer 100 at least 10 more rounds (until Round 110) to switch to RECI.

This translates to the initial higher RECI ratings that correspond to the higher patience values. Again, as in the previous experiment, there are diminishing returns with increasing the patience value beyond a point.

Because receipts are disseminated throughout the community, it becomes easier for RECIs to recognize each other eventually even in the "impatient" case (that is, the case where the patience value equals 1), in which cooperation levels start out low.

At Round 600, the rating for the specific values of the client-server repository sizes is practically the same for all patience values. However, *some* patience is required (that is, patience cannot be zero).

6.2.3 Experiment A-3: The effect of the community growth rate

Table 6.3 Simulation parameters for Experiment A-3

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every 2 rounds (until Round 200)
	1 new peer joins every round (until Round 100)
	10 new peers join every round (until Round 10)
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	100% RECI followers
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

With this experiment, we wanted to see if the community growth rate affects the rating that RECI attains. In Figure 6.5, we plot three lines, showing, from top to bottom, how the rating of RECI changes as rounds progress for growth rates that correspond to:

(1) 1 new peer joining every 2 rounds (top line—Line 1)

(2) 1 new peer joining every round (middle line—Line 2)

(3) 10 new peers joining every round (bottom line—Line 3)

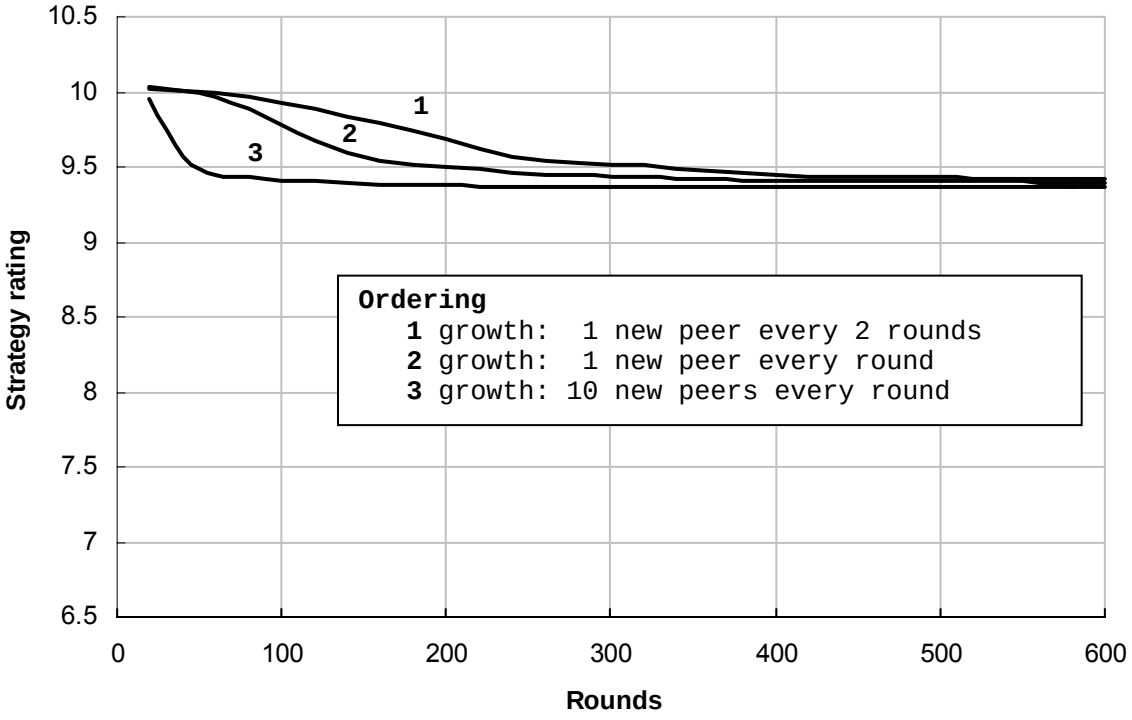

Figure 6.5 The effect of growth rate. For three different community growth rates, the RECI strategy attains a similar rating as rounds progress.

Because all the newly joined peers will be altruistic pre-RECIs until they achieve 10 successful consumptions, this means that if all the peers join the community quickly—corresponding to Line 3 where 10 new peers join every round—then, by Round 10, all 100 peers will have joined the community. At least 10 additional rounds beyond that point will be required for all peers to complete their 10 successful consumptions (see the analysis in the previous experiment), at which point no more altruistic pre-RECIs that cooperate to the full will remain. This is why the rating of RECI follows the particular ordering for the three different community growth rates presented here. The conclusion is that the growth rate does not affect the value around which RECI's rating stabilizes at Round 600.

6.2.4 Experiment A-4: Tracking receipts

Table 6.4 Simulation parameters for Experiment A-4

Parameter	Value
Maximum number of peers	100
Community growth rate	1 new peer joins every round (until Round 100)
Server repository size (receipts)	500, 1000—see experiment
Client repository size (receipts)	10, 200, 500—see experiment
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	100% RECI followers
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

In this experiment, we wanted to track how receipts circulate through the system. For this, we ran three simulations up to Round 500. At that point, we stopped the simulation and examined the contents of all (100 total) server repositories. In Figure 6.6, we plot on the x-axis the receipts generated during a simulation run, starting from the newest (to the right) and moving towards older ones (to the left). On the y-axis, we plot the number of peers/teams that store the particular receipt in their server repository. Time moves from left (past) to the right (future). To the right-most side of the x-axis is the receipt that was the last one generated during the simulation run. This last receipt was, at the end of the simulation (Round 500), stored in only one server repository—the repository of the contributor in the transaction as a result of which this last receipt was generated. Because we ran no additional rounds of simulation after that, this receipt did not have the chance to be copied to other peers through gossiping.

Viewing the scatter plot from right to left, we can see a slow increase in the number of peers that store a particular receipt. This happens as a direct result of the gossiping protocol. The trend is for an exponential increase: as one peer learns of a new receipt through the gossiping protocol (when acting as contributor) he also stores it in its client repository (through the update procedure—see Section 5.2). Then, he disseminates it to other peers when he visits them as a prospective consumer and uses the gossiping protocol.

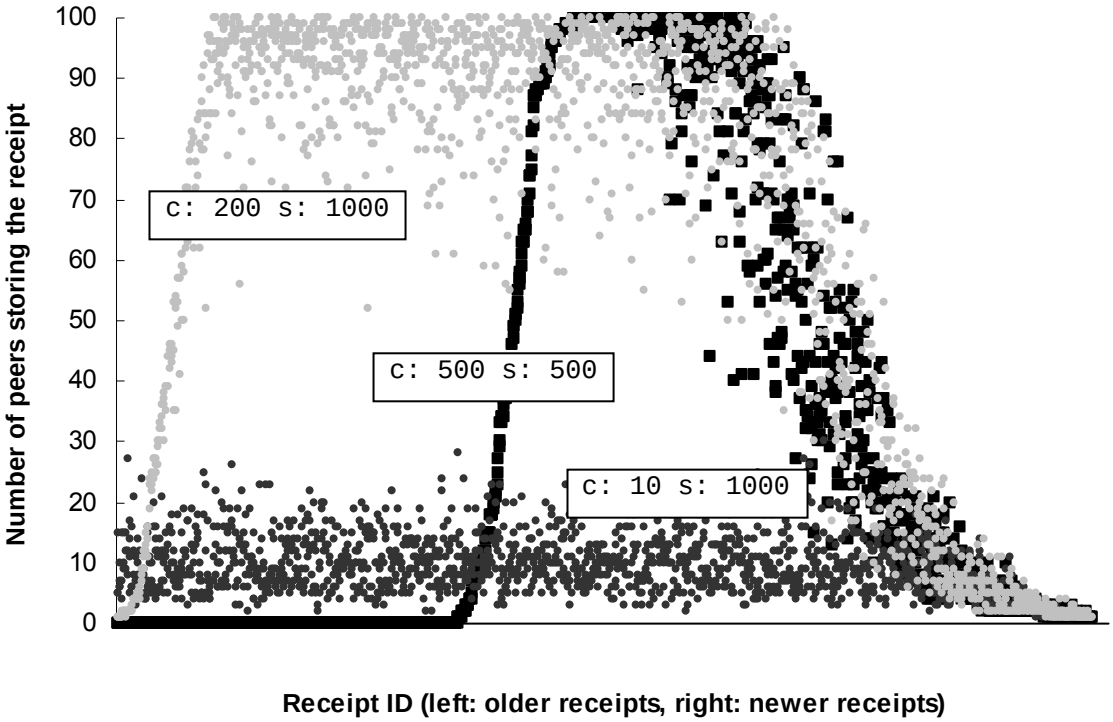

Figure 6.6 Tracking receipts. Depending on the sizes of the client and server repository, the receipts are spread throughout the community via the gossiping algorithm, following different patterns. (The receipt ID is the value that the simulator assigns to IDs internally, as they are created.)

For simulations "c: 500 s: 500" and "c: 200 s: 1000" (corresponding to the black and light-gray colored plots that represent, respectively, the case with 500 receipts in both the client and server repository, and the case with 200 receipts in the client and 1000 receipts in the server repository), many receipts are found in the repositories of almost all the peers in the community. The difference in the width of the topmost part of the "c: 500 s: 500" plot and the "c: 200 s: 1000" plot is because, in the case of the "c: 500 s: 500" plot the server repository size is only 500 receipts (compared to 1000 for the other). As time progresses and newer receipts replace the old ones, older receipts start to disappear from the system. These receipts are relatively newer when the server repositories store only 500 receipts compared to when they can store 1000. (Note that receipt replacement follows a FIFO ordering—see Section 5.2.)

In the third scatter plot ("c: 10 s: 1000," which corresponds to a small client repository of 10 receipts) we see that no receipt can be found in more than 30 peers.

6.2.5 Experiment A-5: Receipt chain lengths

Table 6.5 Simulation parameters for Experiment A-5

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	1, 5, 10, 50, 100, 200—see experiment
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	100% RECI followers
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

In this experiment, we wanted to examine the lengths of the receipt chains, which form the paths of debt that the maxflow algorithm discovers. The minimum chain has a length of one, and is a chain that describes direct debt from Peer P to Peer C. A chain of length two connecting Peer P to Peer C contains one intermediate peer (that is, it is of the form, P→X, X→C), and so on for chains of greater lengths. Examining the chain lengths gives us an indication as to what degree multi-way exchanges (that is, exchanges that involve more than two peers in a reciprocity "ring") are in fact occurring.

Indeed, for the simulations that we conducted with 100 peers and a server repository size of 1000 receipts, the average chain length varied between 2.6 and 3.1. More interestingly, however, as Figure 6.7 shows, the *maximum* chain length (as it was after 500 rounds of simulation) varied greatly depending on the size of the client repository.

Starting with a client repository size of 1, the maximum chain length was, on average, 20.8. (We also plot the average maximum chain length and the 95% confidence interval from 5 sample runs.) It then drops and stabilizes for client repository sizes above 200, where the maximum chain length is usually 6 or 7 (in that respect, the average maximum chain length—around 6.2—and the corresponding confidence interval for these last two points are artificial).

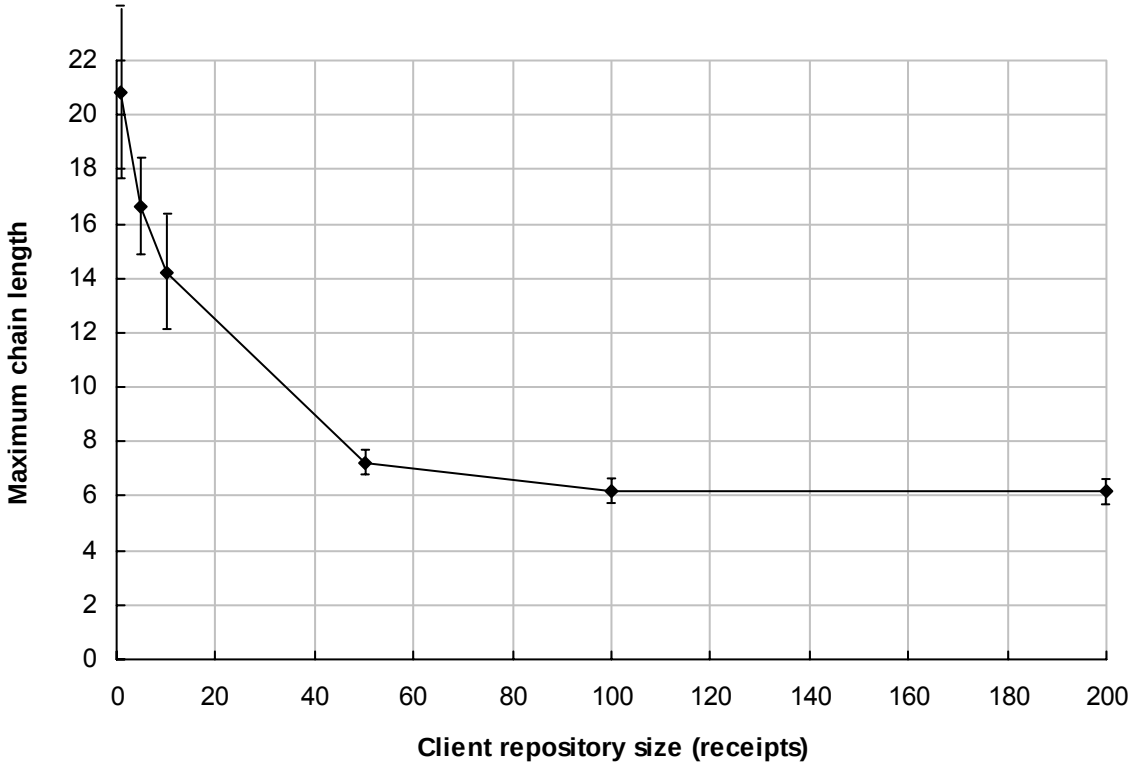

Figure 6.7 Maximum chain length. This represents the maximum receipt chain length (as detected when running the maxflow algorithm—it is the hop-count from the contributor to the consumer) detected after 500 rounds of simulation for various client repository sizes. Small client repository sizes can cause the detection of multi-way exchanges involving more than 20 peers in an exchange ring.

The sudden drop of the maximum chain length as the client repository increases in size can be explained as follows. If the consuming peers bring with them many receipts in their client repositories, then these receipts will in fact be organized in such a way so that trees of receipts are formed, with the consumer as the root of the tree. This happens as a byproduct of the gossiping protocol and the recursive procedure we describe next. When a consumer C shows a prospective contributor P his receipts, Peer P will contribute service only if he can detect himself as a leaf in a tree of receipts rooted at C. This is another way of looking at one component flow that will be part of a non-zero result for the maxflow(P→C). If Peer P detects himself and contributes service, Peer C will issue a new $C \rightarrow P$ receipt, which will effectively connect C's tree as a sub-tree to an equivalent tree in Peer P (because this new receipt and the aforementioned tree will now be stored at Peer P—the tree will be stored there because of receipt merging). See also Figure 6.8.

Note that, with this tree, Peer P can potentially consume from any peer where Peer C could consume before (such as Peer X in Figure 6.8). These peers owed a debt to Peer C

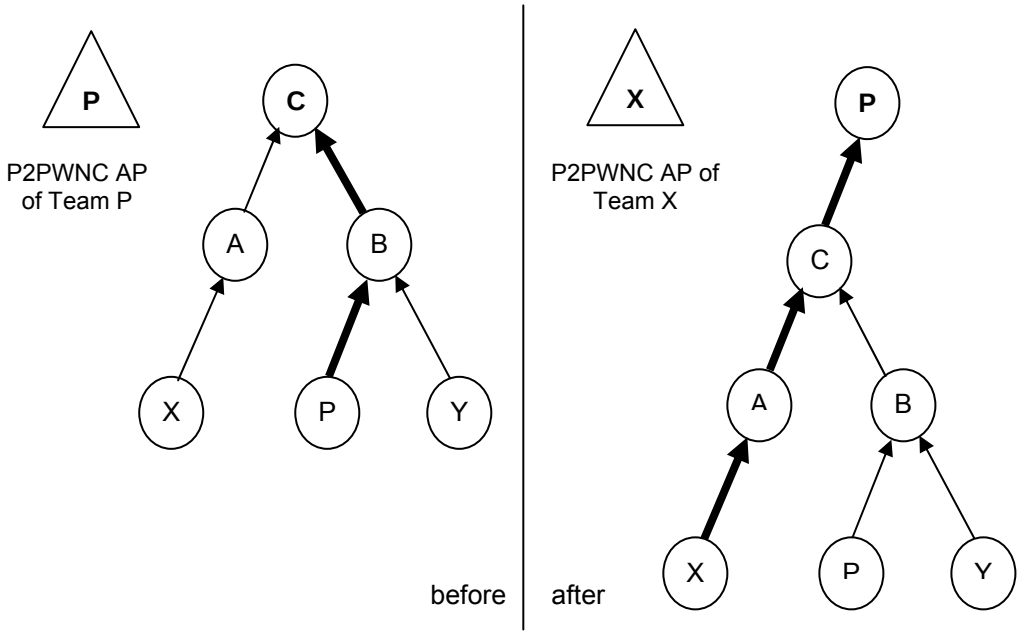

Figure 6.8 Detecting non-zero maxflow from P to C is equivalent to P detecting itself in a receipt tree rooted at C. If P contributes service, C will issue an new C→P receipt which may later enable Peer P, as a consumer, to consume from anywhere that C could consume before (such as Peer X above).

directly or indirectly, and now they owe a debt indirectly to Peer P, because Peer C owes a debt to Peer P. In fact, however, because receipts are deleted from the repositories in the process, this analysis is only a first-order approximation of how server and client repositories evolve.

Nonetheless, we see that with large client repositories, the aforementioned tree of C will have several intact branches, compared to a corresponding tree if the client repository is smaller (assuming the server repository size is the same in both cases). If Peer P owes directly or indirectly to any node in that tree, then Peer P will also owe a debt to Peer C. The more distinct peers appear in this tree, and the more intact are the branches, therefore, the higher the probability of this happening, and therefore Peer P will be able to detect several relatively low-hop paths to Peer C, enough to "congest" all his outgoing edges in the graph as he "pushes" additional flow during the maxflow computation. On the contrary, with smaller client repositories, P does not have the opportunity to find debt over such tree-based low-hop paths to Peer C, and Peer P will also detect longer paths while trying to push all possible flow from his outgoing edges during the maxflow computation.

6.2.6 Experiment A-6: Preferential visitations

Table 6.6 Simulation parameters for Experiment A-6

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	100
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Preferential visitations"—see experiment
Strategy mixture	100% RECI followers
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

With this experiment, we wanted to see how the reciprocity algorithm performs under a different mobility model from the "perfect matching" we used so far. In this model, which we call "preferential visitations," a peer prefers to visit peers that are "nearby" in an ID space that we have defined. More specifically, peers prefer to visit peers that are no more than one hop away in this ID space. Peers obtain IDs as they enter the community, starting from Peer 1 with ID 1 and continuing up to, and including, Peer 100 with ID 100. In this model, Peer 5 prefers to visit Peer 4 and Peer 6. Peer 6 prefers to visit Peer 5 and Peer 7, and so on. (For the corner cases, Peer 1 only prefers to visit Peer 2, and Peer 100 prefers to visit Peer 99—that is, the ID space does not wrap. In addition, in the beginning of the simulation, when the community still grows, the scheme as described still applies—we replace Peer 100 with the peer who has the highest ID at the time. Peers are assigned IDs as they join by increasing a counter.)

This model is simple and attempts to capture the movements of peers who, some of the time, access P2PWNC APs at a few specific locations, and only the rest of the time do they access P2PWNC APs uniformly at random. To capture this in our model we define the "probability to teleport." When this probability is 0, a peer only makes the preferential visits we described above. When the probability is 0.5, a peer spends half his time visiting one of the two neighboring peers, and the rest of the time visiting peers uniformly at random. When the probability to teleport equals 1, the preferential visitations model becomes

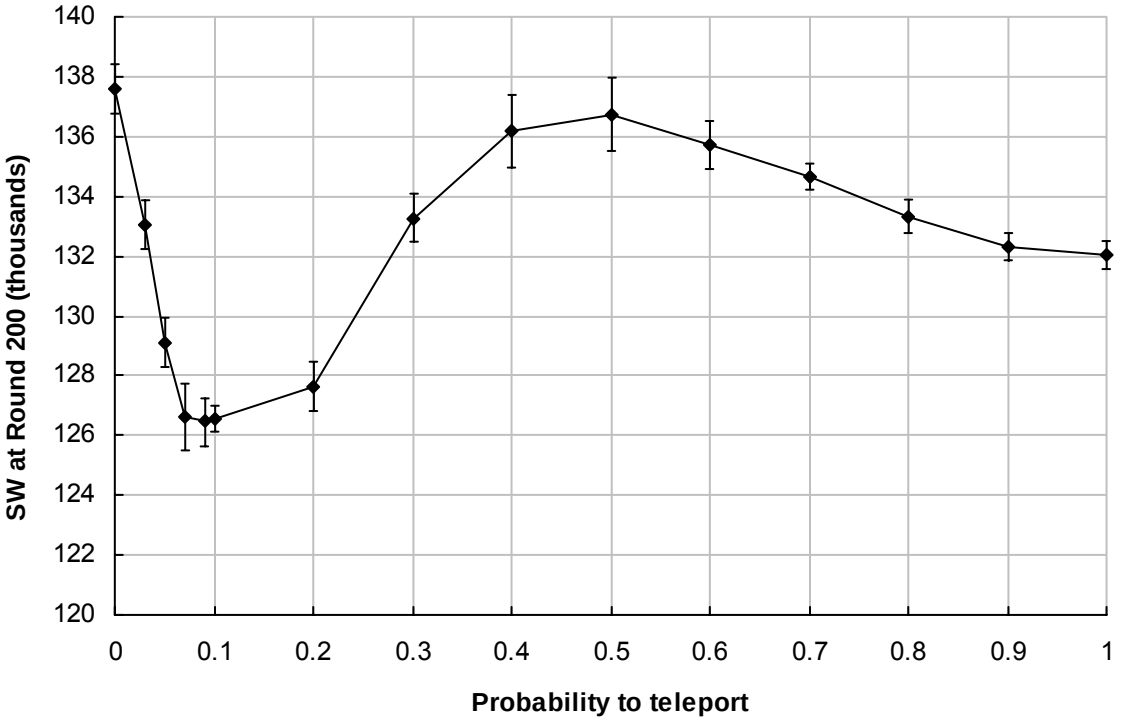

Figure 6.9 Preferential visitations. In this mobility model, the social welfare depends on how a peer's time is divided between the community in general and a few specific locations in particular (expressed by the probability to teleport, which is the fraction of time a peer spends visiting the community in general).

similar to the perfect matching model, however not identical: In the perfect matching model every peer has exactly one opportunity to consume and exactly one opportunity to contribute per round—in this model, and when the probability to teleport equals 1, every peer still has exactly one opportunity to consume per round, but we do not limit the chances of contribution in the same way. Therefore, it is possible that, during a round, some peers will not have an opportunity to contribute, and that some peers will have more than one opportunity to contribute.

In Figure 6.9, we plot the average social welfare (with 95% confidence intervals) at Round 200 for a community of 100 peers (see also Table 6.6). For the right-most value of 1.0 on the x-axis, the social welfare at Round 200 is approximately 132,000. This value represents a median for this set of simulations. For other values of the probability to teleport, the average social welfare at Round 200 is either below 132,000 (going down to approximately 126,000), or above 132,000 (going up to approximately 138,000). This behavior can be explained as follows. For very small values of the probability to teleport, peers effectively form small groups in which a neighbor visits his neighbors, and his neighbors

visit him. Thus, in these groups, the debt that is created is direct (one-hop away) and it is also symmetric in the long run. For example, in the special case where the probability to teleport equals zero, each peer will only have direct debt to his two neighboring peers, which is easy to detect (we have already discussed this in the two-peer example mentioned in Section 6.2.1).

As the probability to teleport increases, the social welfare decreases. Since peers spend most of the time with their neighbors, whenever they visit a random peer, chances are that the (indirect) debt of that peer to them will be small, and they will get little or no cooperation from him. In fact, when the probability to teleport is around 0.09 we get the lowest possible social welfare in this experiment. After that, as the probability to teleport increases, social welfare increases, until we reach a local maximum at probability 0.5. There, the time a peer spends visiting is divided equally between his neighbors and the rest of the community. This is not as good from a social welfare perspective because the global maximum is attained when visiting neighbors only, but it is still a local maximum. After that point on the x-axis, as the probability to teleport increases further, we slowly approach the completely random visitations mode, and the peers do not often get the chance to prove direct debt to neighbors. Still, because everybody visits everybody in the community, the gossiping algorithm circulates enough receipts to attain, at probability 1.0, a higher social welfare than at the global minimum (probability to teleport: 0.09).

6.2.7 Experiment A-7: Erase debt

Table 6.7 Simulation parameters for Experiment A-7

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	100% RECI followers; Peer 100 changes strategy at Round 300 to "erase debt"—see experiment
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

In this experiment, we wanted to see how a specific adversarial strategy, which we call "erase debt," affects the score of the attacker who follows it. We already discussed this attack in Section 5.1.7 where we introduced the need for the r_2 component of the *Subjective Reputation Metric* (SRM) and the GMF algorithm that we defined.

Here, Peer 100 joins the community at Round 100 (we follow the "1 new peer per round" community growth pattern). After this point, it can be seen in Figure 6.10 that his score increases linearly—as is to be expected (see also Figure 6.3). At Round 300, however, Peer 100 decides to switch to the "erase debt" strategy in an attempt to cheat the r_1 component and erase his debt. In practice, this means that if Peer 100 were able to do that, then he could keep his level of contribution the same and recruit many more members in his team, and then have them consume, and then erase the consequences of their consumption as we described in Section 5.1.7. If one member can consume and then erase the consequences of his consumption, then an arbitrary number of members can consume, and the maxflow-based reciprocity algorithm that measures debt in the denominator of r_1 by computing the result of maxflow(C→P) (which would equal zero) would always give the maximum value for r_1 which is 1. This attack was the reason we introduced the GMF-based r_2 component.

To perform the attack, Peer 100 pushes all his "good" reputation to a new (Sybil) ID each time he visits, by connecting this Sybil ID to his old one using a number of unit-weight receipts, which is a parameter of our experiment. (In a more realistic setting, Peer 100 would connect his old ID to his Sybil ID by issuing one receipt with large-enough weight, enough to carry the flow from C to C_i—see also Section 5.1.7.)

Here, the average maxflow result in the specific community is slightly less than 10 (as seen in the simulator), so 10 fake (unit-weight) receipts are usually enough to carry all of the flow from C to C_i without losses (we will see in the experiment that using 20 fake receipts did not improve or worsen the situation for the attacker). In Figure 6.10, we see that, if he performs the attack with only one fake receipt connecting the two IDs, Peer 100 starts *losing* score. What happens is that the average benefit he receives per round is on average less than 1; that is, his benefit does not offset his cost, which is fixed at 1, paid every time he contributes—and he has one opportunity to contribute in every round.

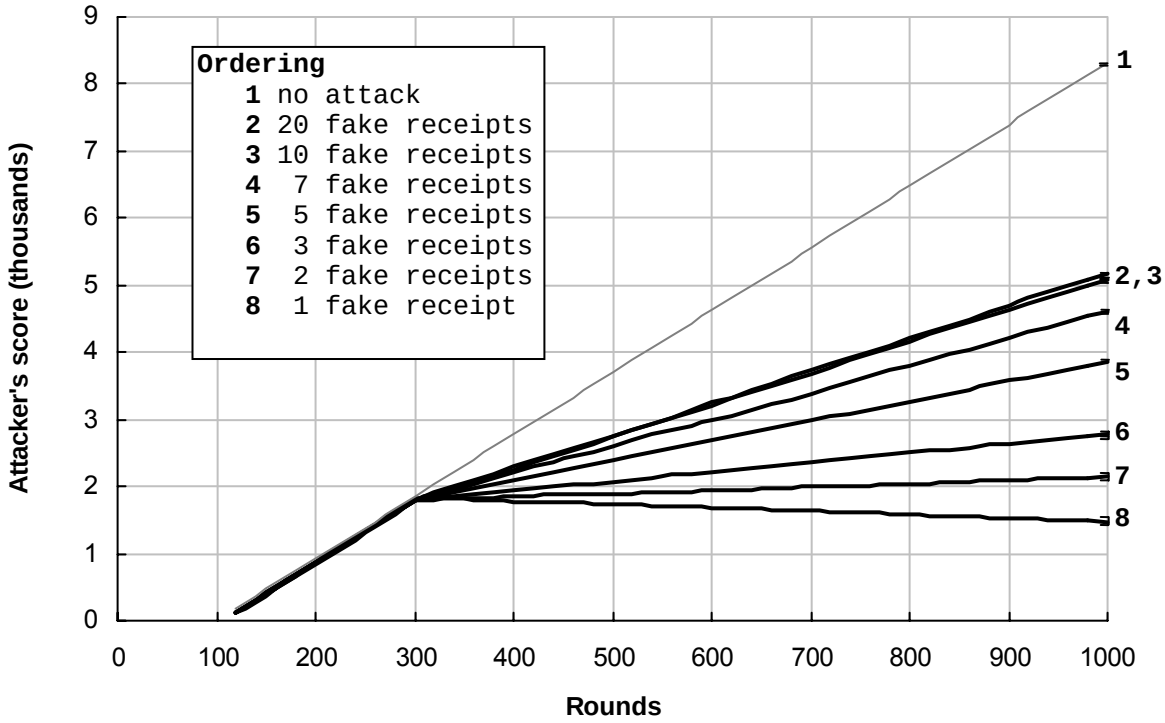

Figure 6.10 Erase debt. To obtain a higher SRM, a peer attempts to "push" all the debt from his ID (see attack description in Section 5.1.7). The GMF component detects this attack.

For the remaining lines, going from two fake receipts to 20 fake receipts we see that he does better. However, in all cases, Peer 100 does worse than if he did not perform the attack (Line 1 in Figure 6.10). We see also that after using 10 fake receipts, there is not much more than he can do by using 20 (or 30, or 40, although these results are not included in this figure). We return to see why the "erase debt" strategy is harmful in the experiment below.

6.2.8 Experiment A-8: Random waypoint mobility

Table 6.8 Simulation parameters for Experiment A-8

Parameter	Value
Maximum number of peers	100
Community growth	All 100 peers are present at Round 1
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Random waypoint"
Strategy mixture	100% RECI followers; in the second part, Peer 100 switches to various strategies at Round 200—see experiment
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

In this experiment, we wanted to test yet another mobility model in addition to the "perfect matching" and "preferential visitations" that we have seen so far. In the "random waypoint" mobility model, peers move in a more realistic way. The "random waypoint" model that we will present below is the same as the random waypoint model proposed in the literature [84] with a pause time of zero. More specifically, we assume that all peers have exactly one P2PWNC server (WLAN AP) and exactly one P2PWNC client. We arrange 100 WLAN APs in a rectangular area with a side equal to 10 Km. The APs are arranged as seen in Figure 6.11: the rectangular area is divided in 100 squares with a side of 1000 meters each, and the AP is placed at the center of the square.

Members move in the following manner. In the beginning of the simulation they are randomly placed anywhere inside the rectangular area. Each member then picks a *destination* in the rectangular area, and a *speed*, chosen uniformly at random between the range of values $[u_{min}, u_{max}]$. Each member then proceeds to his destination using his chosen speed. Once the member arrives at the destination, he requests service from the WLAN AP that is nearest to his current position. We assume that the coverage in the rectangular area is total, so there is always one such WLAN AP. Then, independently of whether his previous request succeeded or not, the member picks another random destination and speed, and starts moving towards it, repeating the process.

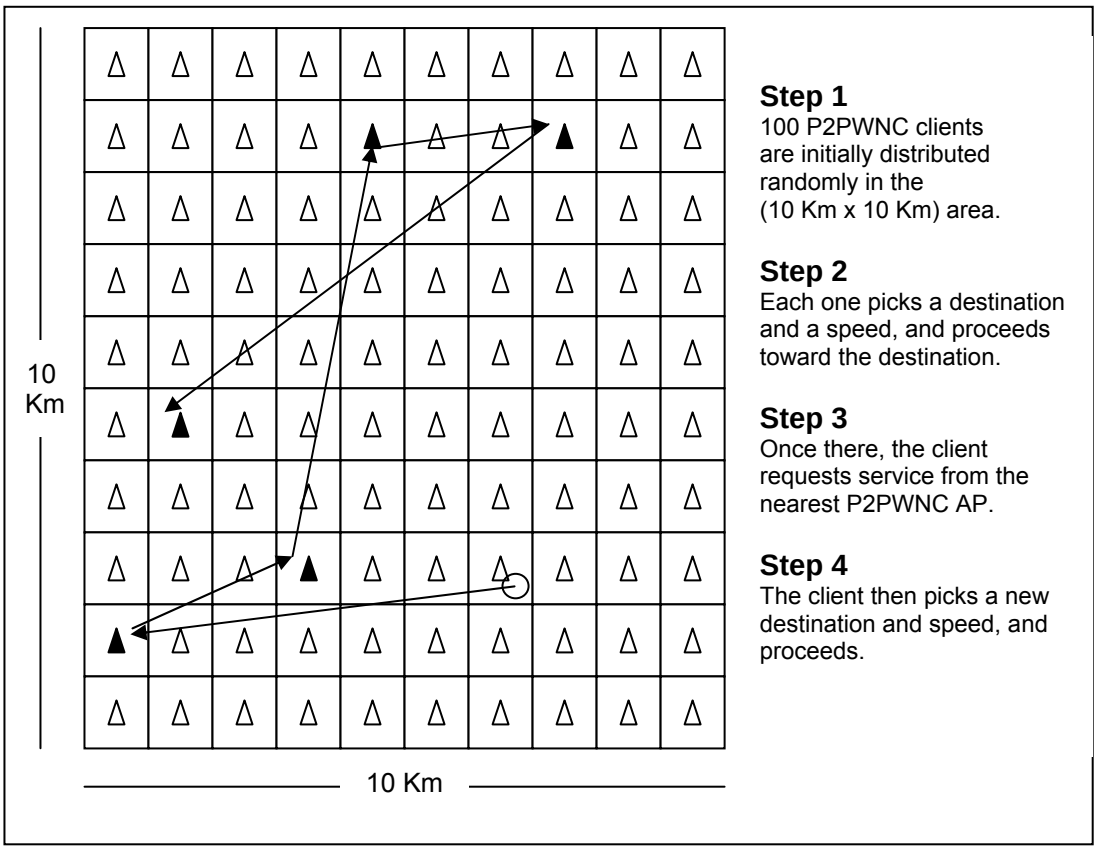

Figure 6.11 The simplified random waypoint mobility model used in Experiment A-8. Figure shows the path followed by one consuming peer. The P2PWNC APs that this peer visited are shown in black. During the simulation, there are a total of 100 consuming peers that follow similar paths. The 10 Km × 10 Km area is divided into 100 squares (side 1000 m each). A P2PWNC AP is at the center of each square.

In Figure 6.12, we plot the average rating of the RECI strategy as rounds progress (a round in the mobility model, according to our original definition, is still a set of matches equal in number to the number of peers—100 in our case; of course, matching is not "perfect"). We use the same minimum speed of 1 m/s but vary the maximum speed that peers can choose from 2 m/s to 20 m/s. Although both ratings stabilize around a value, these values are less than the ratings achieved under "perfect matching" for the same repository sizes (which was 9.40—see Experiment A-1).

In addition, the rating for the higher maximum speed is less than that of the lower maximum speed. This is explained as follows. With the random waypoint model in general, the consumption request rate is not uniform any more. Depending on the speeds and distances chosen, a peer that reaches his destination first will request service earlier than a peer who is still moving towards his own destination. In this way, peers no longer try to

balance their consumption with their contribution. A peer has a chance to contribute whenever he receives a visitor in his area. However, after contributing service, some time may pass without any visitors; at the same time, the same peer (member), acting as a consumer, moves around the area. If the time he takes to arrive to a destination is small, he may make one or more consumption requests before he gets a chance to contribute. This imbalance of consumption to contribution becomes evident in the receipt graph.

This situation is only worse when the maximum speed is higher. In such a situation, a peer may choose speeds that are much faster than the average speed and attempt to consume more times than the current average of the community, something that becomes evident in his rating.

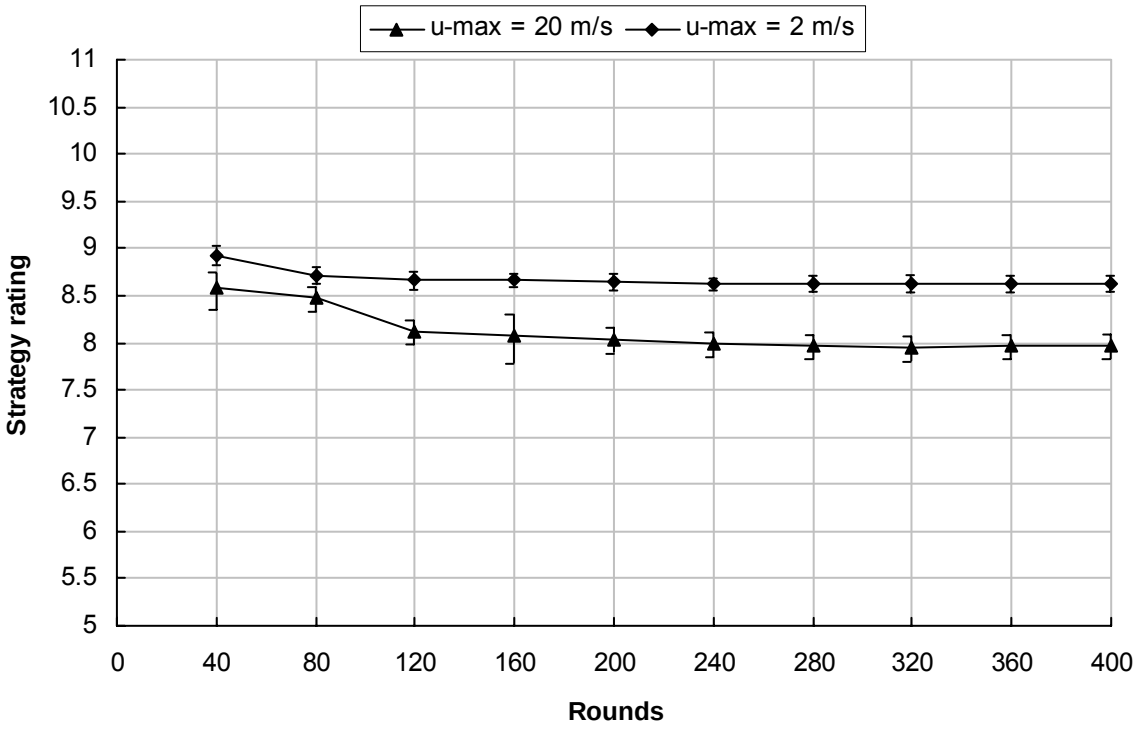

Figure 6.12 The rating of the RECI strategy for two different maximum speeds in our random waypoint model (see also Figure 6.11)

In the second part of this experiment, we let Peer 100 perform the attack that we described in Experiment A-7. This time we perform the attack in stages to show the need for the GMF heuristic compared to a maxflow-based algorithm (similar to the one proposed by [71]), in which only the r_1 component of the SRM would be used. (See also the analysis in Section 5.1.7.)

We can see in Figure 6.13 the baseline rating of Peer 100 when only the r_1 component is used, and the peer is part of a P2PWNC community that uses "random waypoint" mobility with a maximum speed of 2 m/s. The rating stays consistently above 9. We re-ran the experiment, but this time Peer 100, after Round 200, starts to move 10 times as fast compared to the speed dictated by the random waypoint model. That is, even though all the other peers choose a speed between 1 m/s and 2 m/s, Peer 100 does the same only until Round 200. After Round 200, he chooses a speed from the same range, and then multiplies it by 10.

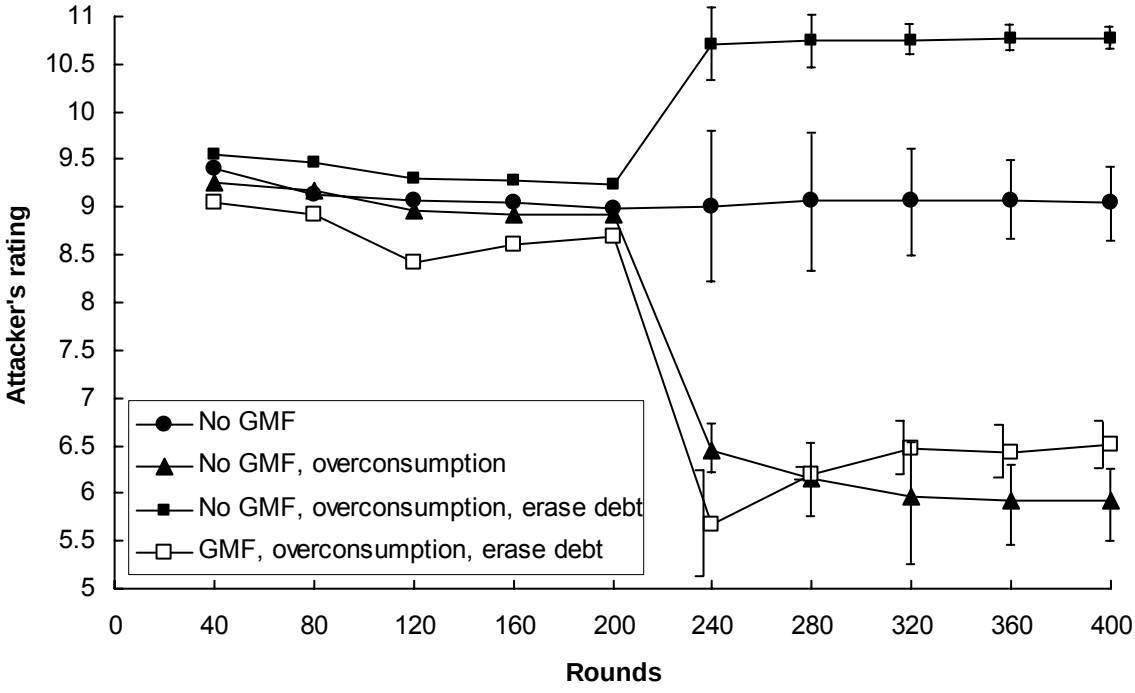

Figure 6.13 The need for the GMF-based r_2 component is shown. Without it, an attacker can start "over-consuming" while at the same time "erasing his debt."

A reciprocity algorithm that uses only the r_1 component is able to detect this over-consumption, because it appears as a higher denominator in the r_1 fraction. The peer is "punished" accordingly by receiving lower rating. Note that because in this mobility model the number of rounds does not correspond to the number of consumption requests of peers, the rating of Peer 100 is measured by taking the total benefit received, minus the total cost incurred, divided by the total number of matches in which he participated as a consumer collecting this specific amount of benefit. Note also that after Round 200 we erase the score

of the attacker gathered until that point, in order to make evident the reduction in average benefit per match after Round 200.

Moving to the third line of Figure 6.13, Peer 100 still changes his consumption behavior after Round 200, but this time he also performs the "erase debt" attack that we described in Experiment A-7. We see now that the r_1 heuristic cannot detect this attack. In fact, it gives Peer 100 the maximum benefit almost every time: Even if one receipt is detected connecting Peer 100 to the contributing peer, the SRM will equal 1 because the denominator of r_1 is zero (and the r_2 component is not used). Of course, the rating is not at the maximum of b_{max} because Peer 100 also contributed service during that time.

Next, in the fourth and final line we re-introduce the r_2 component to the SRM. Peer 100 follows the same tactic, increasing tenfold his average speed after Round 200 and erasing his debt. However, because he uses fake receipts to do that, which push his flow one hop further away from the prospective contributor on the graph, the GMF test of component r_2 detects this, and punishes the peer. Peer 100 has more often than not lower GMF(P→C) than the running average gmf_{avg} that his contributor sees in the community.

We can now see that the GMF test is important. Without that and the r_2 component, a team could perform the following attack: Instead of "increasing the speed tenfold" a team can recruit 10 times more members, effectively increasing the team's consumption rate tenfold, as we did in this experiment. As long as one member of this team can consume and get an average rating above 9, then on average 10 times more team members can do the same as long as they erase the consequences of their action using the "erase debt" attack. However, the GMF heuristic will detect this attack, and all the members will receive less benefit. If we were to look at this team as one peer, then, in absolute terms, the benefit the team receives is greater than that of other peers. For one unit of cost (caused by the single WLAN AP that is owned by the team and is currently operating), the team gets a sum of benefit that is much greater. However, the benefit of a team does not say something about the individual member benefits (which do not simply add up) — the *average* member benefit would be a more realistic descriptor of team success.

If we look at the team members as individuals, every member is worse off and it would be better for each member to start his own team and attempt to balance his consumption with approximately an equal amount of contribution. The experiments so far have shown that the net benefit received will be greater.

6.3 Experiments: Group B

6.3.1 Experiment B-1: ALLC-ALLD mixture

Table 6.9 Simulation parameters for Experiment B-1

Parameter	Value
Maximum number of peers	100
Community growth	All 100 peers join at Round 1
Server repository size (receipts)	1000 (value does not affect experiment)
Client repository size (receipts)	200 (value does not affect experiment)
Patience (consumptions)	10 (value does not affect experiment)
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	ALLDs and ALLCs, starting from 0% ALLD going up to 100% ALLD—see experiment
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

With this experiment, we wanted to introduce the additional strategies that we will use, against which the performance of RECI will be evaluated. Here, we create 11 mixtures consisting of ALLD and ALLC. See also Figure 6.14. We increase the percentage of ALLDs and plot the ratings of the strategies at Round 1000. Since there are exactly 100 peers in the mixture, the number of ALLDs is equal to their percentage. In fact, this experiment validates our simulator and the manner with which it computes the strategy ratings. This is because we can also derive the same results analytically, as we demonstrate below.

For any mixture with n peers, out of which d are ALLDs and $(n - d)$ are ALLCs, and the benefit that ALLCs give to ALLCs and ALLDs whenever they cooperate is b, and the cost of cooperation is c, then, the rating of a single ALLD (which because of uniformity is the rating of all ALLDs) is:

$$rating_{ALLD} = \frac{n-d}{n-1} b \tag{6.2}$$

where n is the number of peers, d is the number of ALLDs, b is the benefit of cooperation.

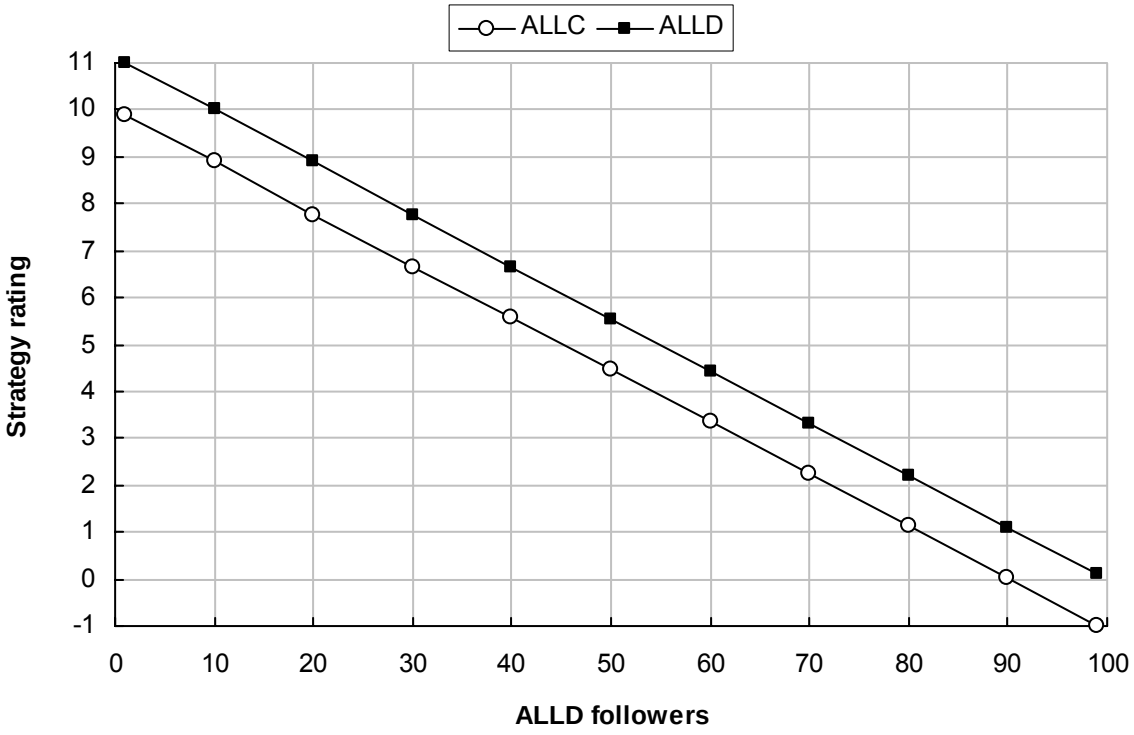

Figure 6.14 The ratings of ALLDs and ALLCs in a mixture of 100 peers as ALLDs increase.

That is, the rating of one ALLD equals the probability that the ALLD follower in question will encounter an ALLC in one of its consumption requests (because of the uniform perfect matching mobility model we use, this equals the fraction of the remaining $n - 1$ peers that are ALLCs), times the benefit he will receive in such a transaction (a transaction which will always succeed because ALLCs always cooperate). In addition, because ALLDs never cooperate, there is no cost component to subtract.

Following a similar analysis, the rating of ALLC is:

$$rating_{ALLC} = \frac{n-d-1}{n-1}b - c \tag{6.3}$$

where n is the number of peers, d is the number of ALLDs, b is the benefit of consuming, and c is the cost of contributing. (The extra (–1) in the numerator compared to Equation 6.2 is there because we focus on one ALLC follower and count all the *other* ALLC followers.)

As an example, when there are 20 ALLDs and 80 ALLCs in the mixture, the rating of ALLD using Equation 6.2 is:

$$rating_{ALLD} = \frac{100-20}{100-1}11 \approx 8.8$$

and the rating of ALLC using Equation 6.3 is:

$$rating_{ALLC} = \frac{100-20-1}{100-1}11 - 1 \approx 7.7.$$

These values were also derived by our simulator (by Round 1000, the values from the simulator approached the analytically derived ones to a precision of 2 decimals).

We can see from Figure 6.14 and the analysis above that ALLDs always perform better than ALLCs. Even though, as ALLDs increase in numbers the rating of each strategy worsens, the ALLD strategy still performs better. For the corner cases, when no ALLDs exist and there are 100 ALLCs in the community, the system reaches the optimal level of cooperation, and the optimal rating which is 10. When there are no ALLCs, ALLD rating is zero because ALLDs do not cooperate with each other. In addition, in a mixture with 99 ALLDs and 1 ALLC, the rating of the single ALLC follower is exactly –1, and the rating of ALLD is 0.11. In the case of a mixture with 1 ALLD and 99 ALLCs, the rating of ALLC is 9.89, and the rating of the single ALLD is 11.

6.3.2 Experiment B-2: ALLC-ALLD-RECI mixture

Table 6.10 Simulation parameters for Experiment B-2

Parameter	Value
Maximum number of peers	99
Community growth	1 new peer joins every round (until Round 99)
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	33 ALLD followers, 33 ALLC followers, 33 RECI followers—see experiment
Evolution	$p_l = 0$, $p_m = 0$ (no learning, no mutation)

In this experiment, we plot the ratings of ALLC, ALLD, and RECI, as they change in 3000 rounds of simulation. In the mixture of this experiment, there are 33 followers of each strategy. We can make two observations by looking at Figure 6.15. First, RECI performs better than both ALLD and ALLC. ALLC performs worse than RECI, but in a predictable way. The difference in rating of approximately 0.34 between RECI and ALLC at Round 3000 is approximately equal to the fraction 33/98. This fraction equals the probability that an ALLD will visit an ALLC in the perfect matching model of this experiment where there are 99 peers in total and 33 followers of each one of the three strategies. On the one hand, RECI always recognizes an ALLD follower (the SRM for that peer is zero) and never cooperates, keeping RECI's costs down. On the other hand, ALLCs cooperate also with ALLDs, thereby increasing their cost. This example illustrates how an unconditional cooperation strategy (ALLC) is more costly than a conditional cooperation strategy (RECI) that detects free-riders, when free-riders are present in the mixture.

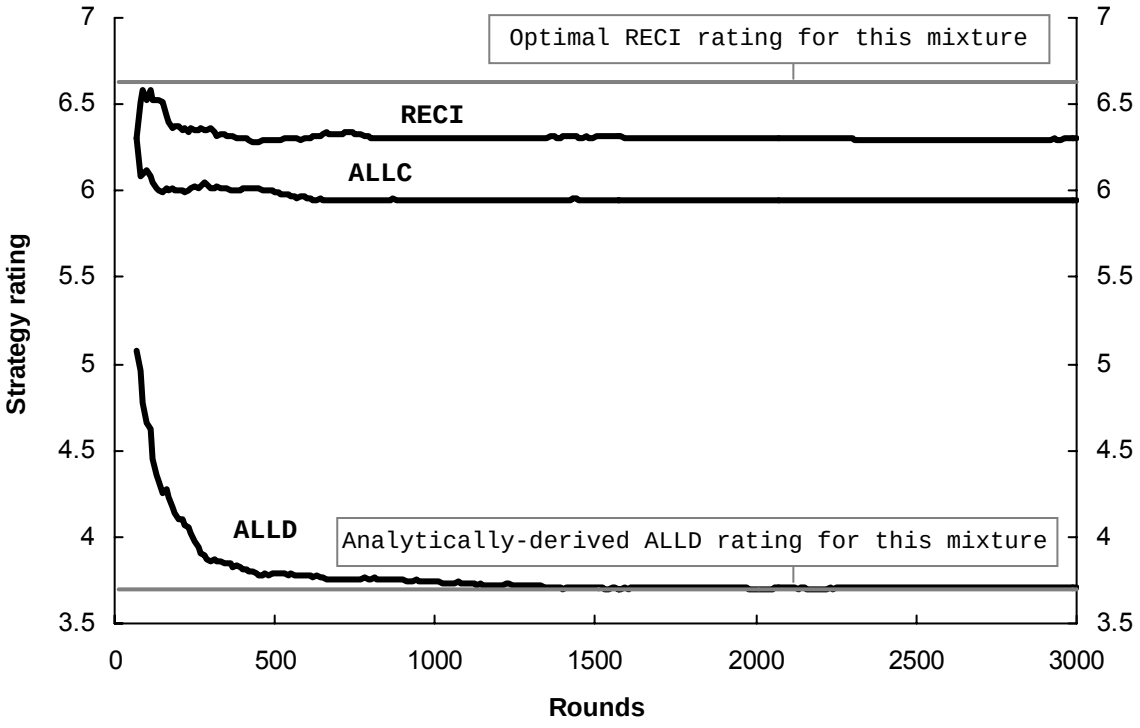

Figure 6.15 ALLC, ALLD, and RECI. RECIs perform better than either ALLC or ALLD. RECI always recognizes ALLD and this is shown here: ALLD rating in the simulation approaches the ALLD rating that is analytically derived based on RECIs that always recognize ALLDs. The same does not apply for RECIs and ALLCs, which fail to cooperate to the full every time, hence the rating that is lower than the optimal.

This difference in rating between RECI and ALLC, and between RECI and ALLD will become important later, when we focus on evolutionary games, in which case the difference in rating in favor of RECI also translates to an increase in RECI followers.

We can make an analysis similar to that of the previous experiment here in order to derive the rating of ALLD analytically. ALLD only receives service from ALLC, so according to Equation 6.2 (if we take into account that RECI will never cooperate with ALLD), the rating of an ALLD follower is 33/98 × 11, which approximately equals 3.70, which is the rating of ALLD followers in this experiment. The initially higher rating of ALLD followers is attributable to the pre-RECI phase of RECIs, where they too cooperate with everyone; during this phase, ALLDs take advantage of pre-RECI patience.

We plot a straight line at rating 3.70 to indicate the analytically derived rating of ALLDs. We also plot a straight line at rating 6.63, which is (approximately, rounding to two decimal places) the value that the rating of RECI *would be* if RECIs cooperated to the full with other RECIs and ALLCs, and never contributed to ALLD (that is, 65/98 × 11 of benefit per round, minus 65/98 × 1 of cost per round). This optimal is not reached, as we have seen in previous experiments (such as Experiment A-1, Figure 6.2).

6.3.3 Experiment B-3: ALLC and ALLD with learning

Table 6.11 Simulation parameters for Experiment B-3

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until round 100)
Server repository size (receipts)	1000 (value does not affect experiment)
Client repository size (receipts)	200 (value does not affect experiment)
Patience (consumptions)	10 (value does not affect experiment)
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture (probabilities)	50% ALLC, 50% ALLD
Evolution	$p_l = 0.2$, $p_m = 0.001$

In this simulation, we wanted to give an example of the power of evolution to change the strategy mixture. We simulate a community with a total number of 100 peers. New peers join the community once every round (until Round 100). Each new peer has

50% probability of following the ALLC strategy and 50% probability of following the ALLD strategy.

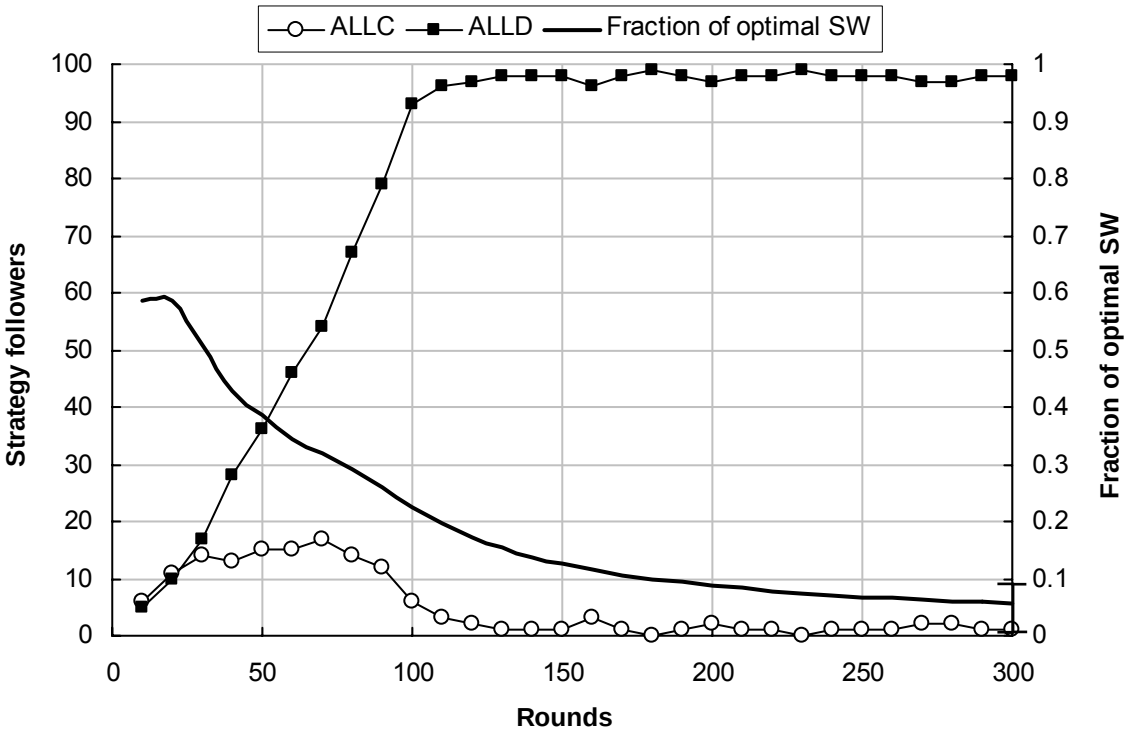

Figure 6.16 If altruism (ALLC) and free-riding (ALLD) are the only available strategies, eventually all will learn to free-ride, bringing cooperation to a halt.

In Figure 6.16, we plot on the primary y-axis the number of followers of each strategy, and on the x-axis the rounds as the simulation progresses. It can be seen from Figure 6.16 that, as peers enter, the ALLD strategy gains more followers than the 50% we probabilistically allow during community growth. These additional ALLDs are recently joined ALLC followers that learn of the existence of a better—in selfish terms—strategy, and switch to following ALLD instead of ALLC. The reason they do that is the higher rating that ALLD achieves, which we already saw (without evolution) in Figure 6.14.

We also plot in Figure 6.16 on the secondary y-axis the social welfare attained (black line). More specifically, we plot the *fraction of the optimal social welfare* attained (see Section 6.1.5). Note that the optimal social welfare is attained when after every cooperation request the contributor cooperates to the full, offering b_{max} benefit to the consumer. Here, we can see that this fraction approaches zero because the ALLD strategy

prevails, and the cooperation level in the community falls to zero (we plot the 95% confidence interval at Round 300; the line itself is from an example simulation run).

This result shows that in such an ALLC-ALLD mixture, the only sustainable equilibrium is 100% ALLD. Moreover (although we do not show it here), this result would be the same for any initial mixture of ALLC-ALLD (even 100-0 ALLC-ALLD), as long as the *possibility* of the ALLD strategy exists through mutation, and as long as these two are the only available strategies in the universe of strategies.

Therefore, even with 0% ALLD initally, when an ALLD follower appears, its rating will be greater that than of ALLCs, and, with a speed depending on the probability of learning, more and more ALLC followers will switch to ALLD, lowering cooperation levels. In that respect, even a 100% ALLC mixture (in which the social welfare attained would be optimal) is *unstable*.

6.3.4 Experiment B-4: ALLC, ALLD, and RAND with learning

Table 6.12 Simulation parameters for Experiment B-4

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000 (value does not affect experiment)
Client repository size (receipts)	200 (value does not affect experiment)
Patience (consumptions)	10 (value does not affect experiment)
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture (probabilities)	33% ALLC, 33% ALLD, 34% RAND
Evolution	$p_l = 0.2$, $p_m = 0.001$

In this experiment, we change the mixture and the set of available strategies compared to the previous experiment, and add the possibility of RAND followers through mutation, as well as a 34% probability of any new entrant to the community to be a RAND follower.

The outcome, as can been in Figure 6.17, is not different from the previous simulation. ALLDs prevail, and the fraction of optimal social welfare approaches zero. Therefore, the RAND strategy does not perform better in evolutionary terms compared to

ALLD, and it cannot stop the invasion of ALLD. Note the local maximum of the fraction of optimal SW after Round 50; at that point ALLC has reached its largest number of followers (19 followers), who then started switching to ALLD or RAND. Eventually most RAND and ALLC followers switched to ALLD.

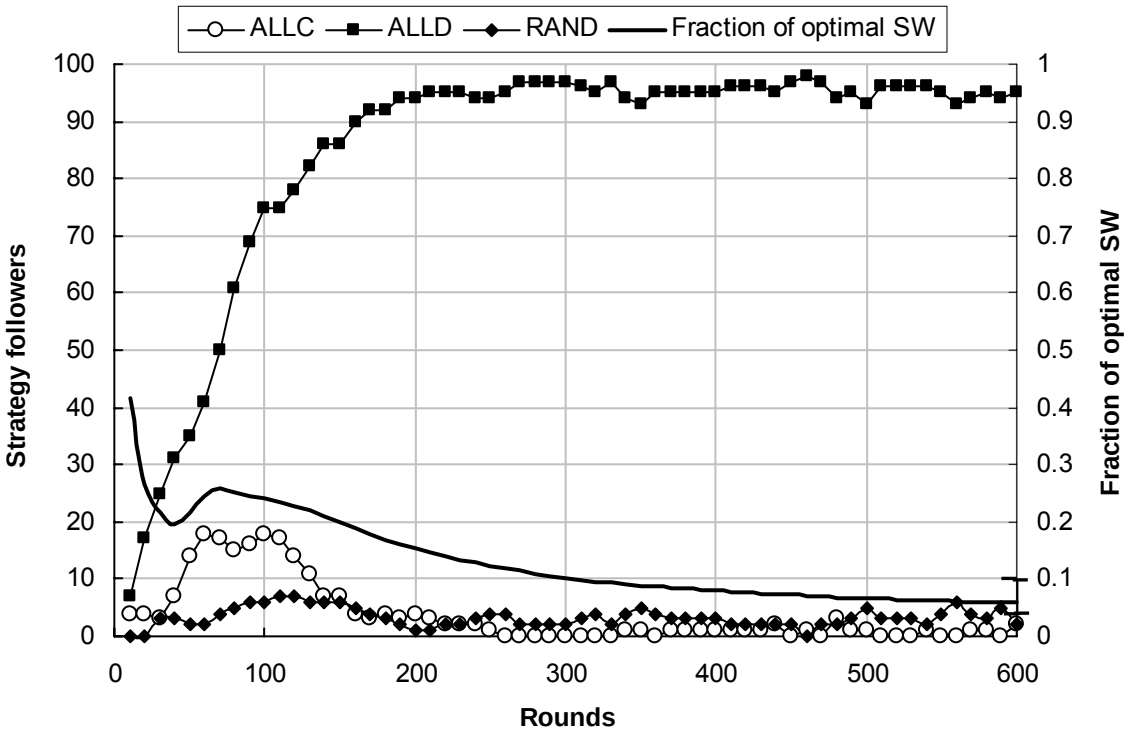

Figure 6.17 If ALLC, ALLD, and RAND are the only available strategies, eventually all learn and switch to ALLD, bringing cooperation to a halt (see also Figure 6.16).

6.3.5 Experiment B-5A: ALLC, ALLD, RAND, and RECI with learning

Table 6.13 Simulation parameters for Experiment B-5A

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture (probabilities)	33% ALLC, 33% ALLD, 34% RAND
Evolution	$p_l = 0.2$, $p_m = 0.001$

In this experiment, we wanted to see if RECI can invade a mixture of strategies if only the possibility of RECI exists through mutation (that is, RECI is part of the set of *possible* strategies, but is not part of the initial mixture of strategies). Here, as new peers join the community, no one follows the RECI strategy. It can be seen, however, that as soon as RECI is "discovered" through mutation, then, *learning* makes peers switch to RECI because of the high ratings it obtains. Thus, RECI can invade an ALLC, ALLD, RAND mixture.

Because of the existence and prevalence of the RECI strategy, the fraction of optimal SW can surpass 0.8 by Round 3000. Here we also see an effect known in evolutionary games as *random drift*, which is inspired by the phenomenon of *random genetic drift* [85]. Because eventually ALLD does not have many followers in the mixture, the ALLC strategy performs the same as RECI (but only because the RECI strategy managed first to "drive away" ALLDs). Therefore, the forces of learning and mutation make random choices between RECI and ALLC, driving them to almost equal representation in the mixture. However, RECI followers in general outnumber ALLC followers exactly because ALLDs reappear occasionally through mutation, exploiting ALLCs. This leads to a temporary increase in ALLD followers, which costs ALLCs more than it cost RECIs (which always detect ALLDs and never cooperate with them).

Note, therefore, the oscillations caused by these periodic limited invasions (and subsequent withdrawals) of ALLD and RAND. (They become more evident in later experiments.) Note also the large confidence interval of Figure 6.18: this is because the invasion of RECI itself may not happen immediately—in the specific simulation run that is depicted on Figure 6.18, RECI percentage surpasses 50% only after Round 500.

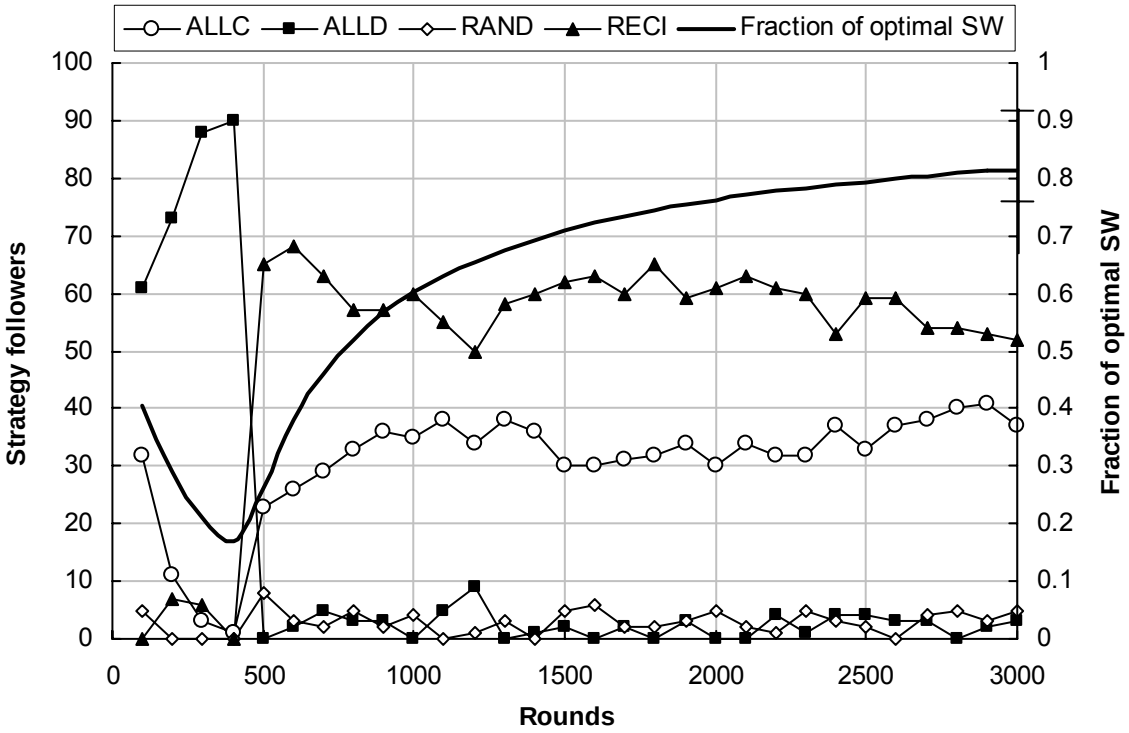

Figure 6.18 Compared to the experiments of Figures 6.16 and 6.17, this time RECI is one of the available strategies. It is not present in the original mixture, appearing only through random mutations. Nevertheless, once it does, it spreads in the community, combating ALLDs and RANDs, and allowing ALLCs to persist alongside.

6.3.6 Experiment B-5B: ALLC, ALLD, RAND, and RECI with fast learning

Table 6.14 Simulation parameters for Experiment B-5B

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture (probabilities)	33% ALLC, 33% ALLD, 34% RAND
Evolution	$p_l = 1.0$, $p_m = 0.001$

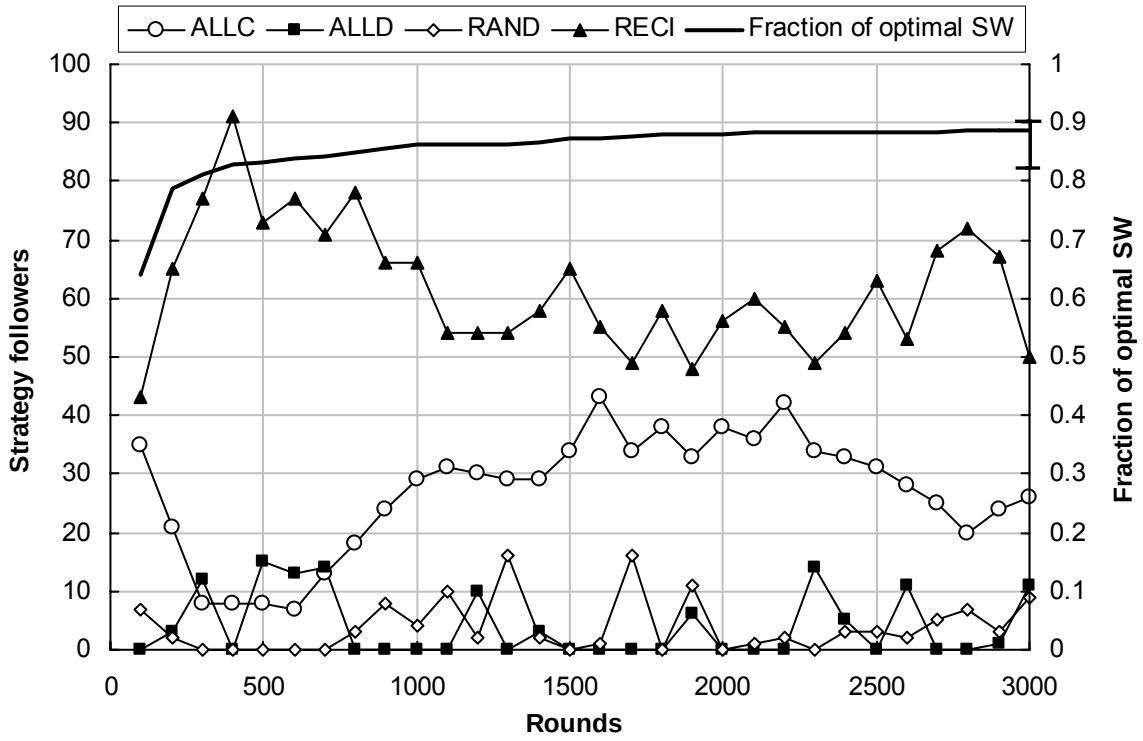

Figure 6.19 Compared to the experiment depicted in Figure 6.18, this time learning probability is 5 times higher. The community attains higher social welfare sooner than before, as peers switch to RECI faster. Note the constant invasion attempts by ALLD and RAND, which are more successful than in Figure 6.18.

This experiment is the same as Experiment B-5A, with one difference. The probability of learning is 1.0 (compared to 0.2 for B-5A). This has two effects. First, the fraction of optimal SW can now be seen to approach the value of 0.9 sooner than before.

Second, there are "spikes" in Figure 6.19 that did not exist in Figure 6.18. This is because of the following. As the evolutionary game progresses, there are differences in rating that can be attributed to chance. For example, an ALLD follower can appear through mutation and end up requesting service from an ALLC follower, who will cooperate to the full. This gives the maximum benefit to the ALLD follower, and, if no other ALLDs exist, a rating of 11 to ALLD (for the benefit function of this experiment). A rating of 11 in the previous round is enough to make a lot of peers switch to the ALLD strategy in the following round. The spikes can be seen as constant "invasion attempts" by ALLDs, which are not allowed to reach a critical mass though: ALLD does not manage to amass more than 14 followers in any such attempt in this simulation run. The same applies to RAND invasion attempts: RAND does not manage to amass more than 16 followers in any such attempt in this simulation run.

Note that we have established from additional simulations (not shown here) that RECI will invade for other, smaller, learning probabilities. For example, we had successes with $p_l = 0.04$ similar to the ones we saw here (with $p_l = 1$) and in the previous experiment (with $p_l = 0.2$).

6.3.7 Experiment B-5C: ALLC, ALLD, RAND, and RECI with fast learning (long)

Table 6.15 Simulation parameters for Experiment B-5C

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until Round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture (probabilities)	33% ALLC, 33% ALLD, 34% RAND
Evolution	$p_l = 1.0$, $p_m = 0.001$

This experiment is the same as Experiment B-5B, with one difference: We have allowed this simulation to run to Round 20,000 (see Figure 6.20). This way, we take a longer view of the system and see if our remarks from Experiment B-5B still hold. Indeed, all the characteristics of the previous experiment appear over longer time periods, such as the random drift between ALLCs and RECIs and the fact that the fraction of optimal SW has stabilized around 0.90 for these parameter values. In Experiment A-1, Figure 6.2, we had seen that for 200 receipts in the client repository and 1000 receipts in the server repository, when RECI is alone in the mixture, it achieved a rating of 9.4 (which in a RECI-only mixture would correspond to a fraction of optimal SW equal to 0.94). The difference between 0.94 in that case and the 0.90 average for this case can be attributed to the continuous invasion attempts by ALLD and RAND, which are unsuccessful, but, nevertheless, are frequent enough to deny ALLCs and RECIs some of the benefit they would give to each other if they were alone in the mixture.

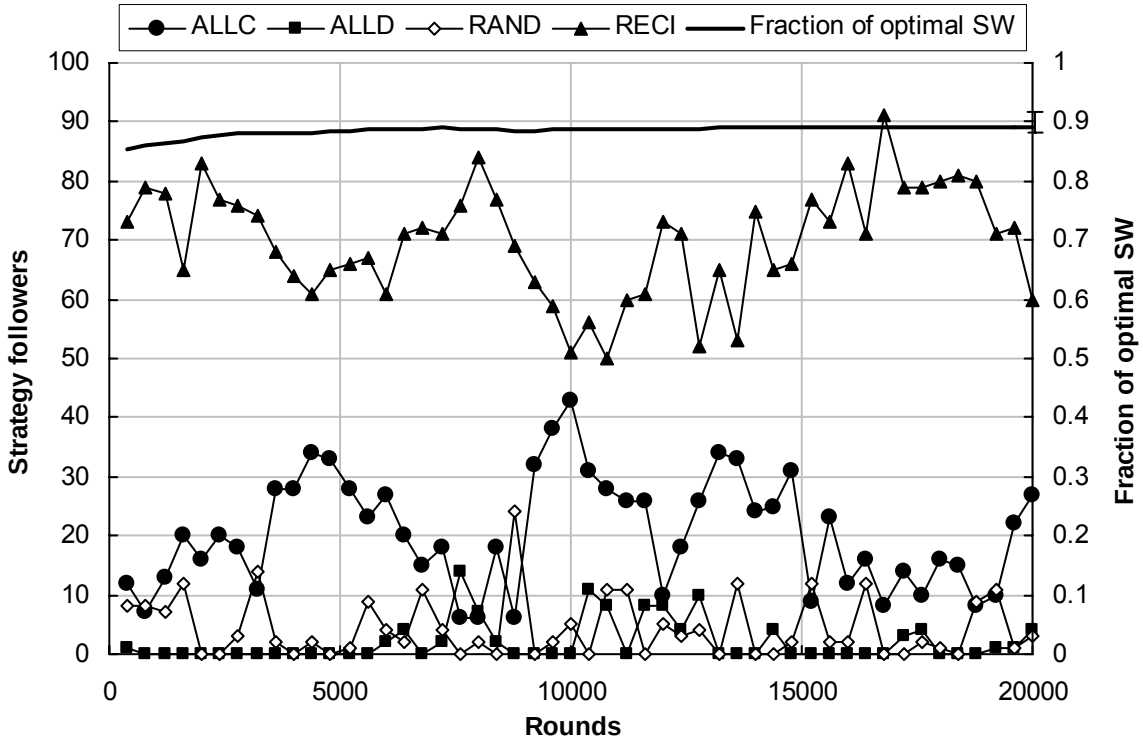

Figure 6.20 In the long term, oscillations are once again observed, similar to Figure 6.19.

6.3.8 Experiment B-6: Withstanding invasion, and larger P2PWNC communities

Table 6.16 Simulation parameters for Experiment B-6

Parameter	Value
Maximum number of peers	100, 1000—see experiment
Community growth	1 new peer joins every round (until Rounds 100, 1000 respectively for the 2 community sizes—see experiment)
Server repository size (receipts)	1000, 6000—see experiment
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	$b_{max} = 11$
Mobility model	"Perfect matching"
Strategy mixture	100% RECI
Evolution	$p_l = 0.2$, $p_m = 0.001$

In the two experiments that follow, we wanted to explore two things: whether RECI can resist invasions from the set of three other possible strategies from the universe of possible strategies: ALLC, ALLD, and RAND. For this, in the two experiments that follow, the mixture starts out 100% RECI.

Our second objective was to test larger communities. So far, we limited ourselves to communities of 100 peers. We wanted to see if we can keep the client repository relatively small, and fixed at 200 receipts, but increase the number of peers in the community. We expect, according to experiment A-1, that the size of the client repository will affect the social welfare if the community grows to larger numbers, and we wanted to see how much. More specifically, we expected social welfare to reduce as the community grows larger and the client repository size stays the same.

In the first experiment (Figure 6.21) the community grows to 100 peers, the client repository size is 200 receipts, and the server repository size is 1000 receipts. We see again that even though RECI starts at 100% in the mixture, there is random drift and we end up with a mixture of RECI and ALLC with around 50% representation from each of the two strategies. Of course, the occasional invasions by ALLD keep ALLC rating lower than that of RECI. However, this does not affect the fraction of optimal SW: indeed, at Round 5000, it is more than 0.91.

In the second experiment (Figure 6.22), the community grows to 1000 peers. We see again the same general behavior, and at round 10,000 the fraction of optimal SW is 0.82. We also see that RAND followers perform better in general because of the larger community size. Thus, we see that we can indeed maintain the same client repository size over a range of community sizes that spans an order of magnitude (with appropriate adaptation of the server repository size). RECI can withstand invasion, but the cost of increasing the community size and keeping the same client repository size is a reduction in social welfare.

Although by changing the server repository sizes we did not control all the parameters of the experiment compared to the previous, we claim that the server repository size does not represent a practical problem, and that its size can be increased at minimal cost in a real-world deployment—that is why we concentrate on the client repository size.

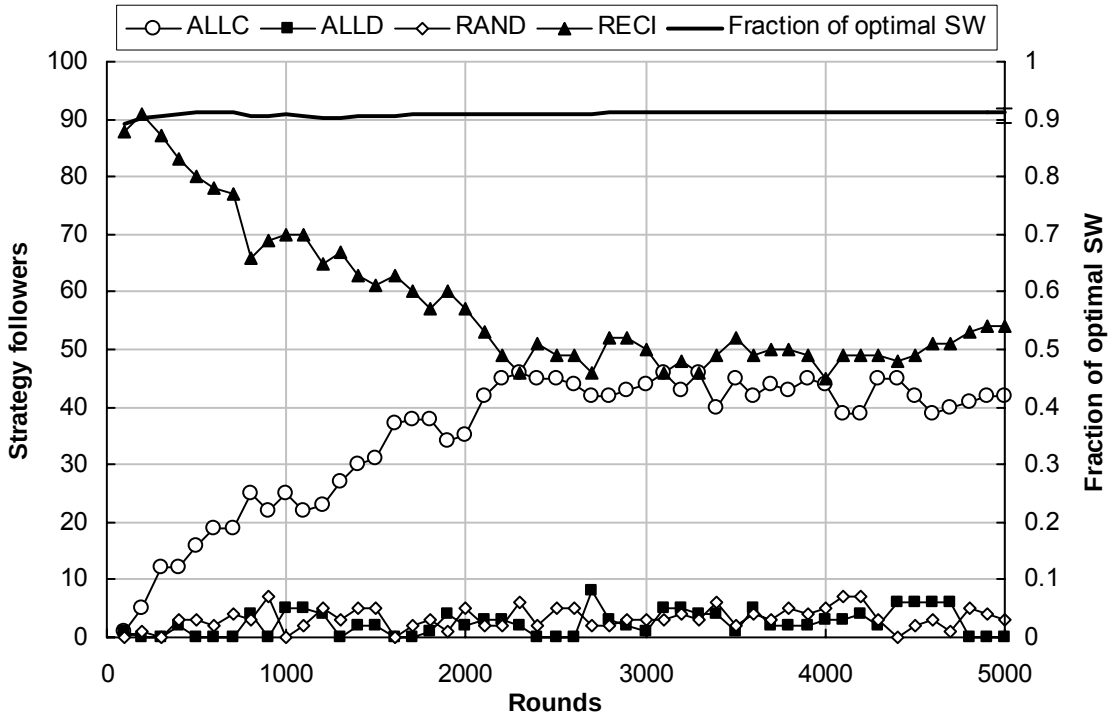

Figure 6.21 Starting from a 100% RECI mixture, ALLCs "invade" because of random drift. This has no discernible effect on the fraction of optimal SW, because RECIs do not allow ALLDs and RANDs to increase their numbers even though ALLCs will cooperate with ALLDs.

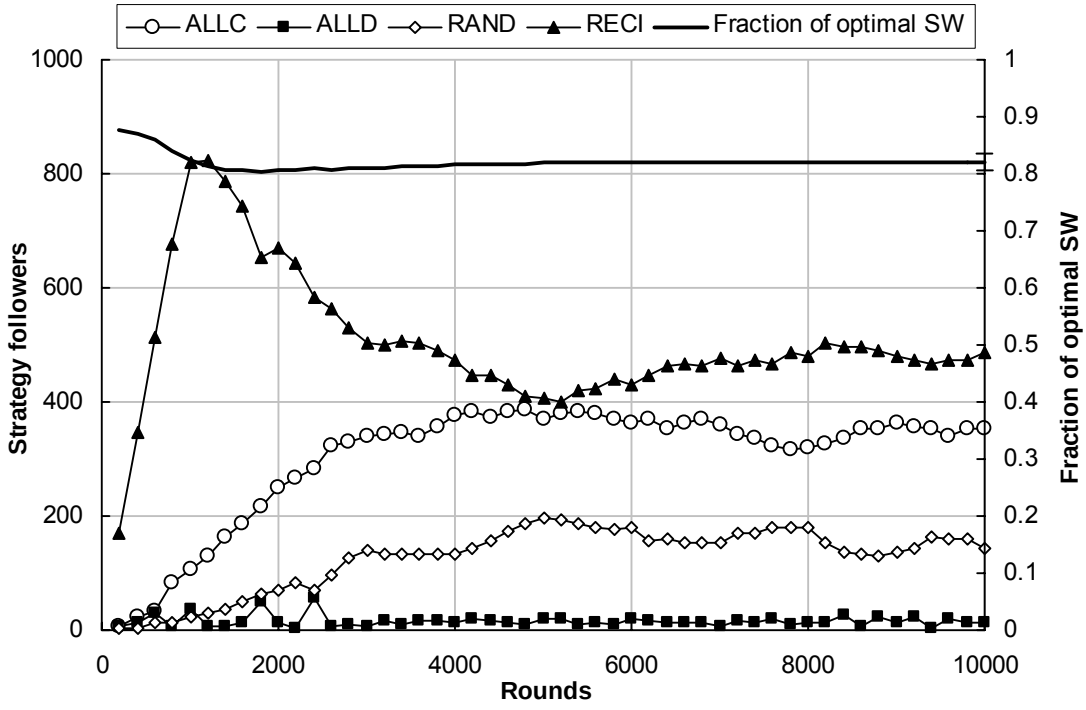

Figure 6.22 The smaller client repository compared to Figure 6.21 causes a reduction in social welfare. Nevertheless, the efficiency loss is small. Notice that RANDs persist here because their rating is now closer to the rating of RECI compared to Figure 6.21.

6.3.9 Experiment B-7: Variable b_{max}

Table 6.17 Simulation parameters for Experiment B-7

Parameter	Value
Maximum number of peers	100
Community growth	1 new peer joins every round (until round 100)
Server repository size (receipts)	1000
Client repository size (receipts)	200
Patience (consumptions)	10
Benefit function	b_{max} = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 20—see experiment
Mobility model	"Perfect matching"
Strategy mixture (probabilities)	25% ALLC, 25% ALLD, 25% RAND, 25% RECI
Evolution	$p_l = 0.2$, $p_m = 0.001$

In this experiment, we wanted to show how the value of the P2PWNC service, as expressed by the b_{max} / c ratio affects the fraction of optimal social welfare attained. To do this, we plot in Figure 6.23 on the x-axis various values for b_{max} (which, as we fix the cost c to equal one, is the same as the value of the ratio), and we plot the average and the 95% confidence interval for the fraction of optimal social welfare at Round 600.

Let us focus on two examples: if b_{max} is 11 then the optimal SW per match equals 10. So for the point at b_{max} = 11, a fraction of 0.823 means that the SW per match was 8.23. For the point at b_{max} = 3, a fraction of 0.13 means that the SW per match was 0.26 (this is equal to (3–1) × 0.13). That is, we are not interested in the actual values of SW, because these depend on the value of b_{max}—we are only interested in what fraction of the optimal SW this represents.

It can be seen that for values of b_{max} less than 7, the community is not doing well. The P2PWNC service is not valuable enough in the sense that adherence to the RECI strategy does not bring substantial increase in benefit compared to other strategies, and, therefore, the evolutionary forces drive RECI followers away from RECI, towards ALLD and RAND, and the fraction of optimal SW drops. After b_{max} exceeds 7, and for values beyond 11, we see that there is no appreciable gain. This means that, for communities where b_{max} exceeds 7, the fraction of optimal SW attained will be close to the values we encountered in the previous experiments.

The large confidence intervals around values 4, 5, and 6 indicate that a transition is happening. For these values, RECI may or may not prove better than the other strategies in the long run; this is why the fraction of SW attained varies greatly. For the b_{max} values 2 and 3, the P2PWNC service is simply not valuable enough, and this, coupled with the fact that RECI may not recognize another RECI follower because of lack of information, as we have seen already, makes it an unprofitable strategy compared to ALLD for this mixture.

We believe that 7 (and 10, and 11) is a very conservative estimate for the benefit-to-cost ratio of the P2PWNC service, even taking into account all the cost factors that we identified before. Therefore, we can support that our previous results hold irrespective of the specific value (b_{max} = 11) that we selected for conducting them.

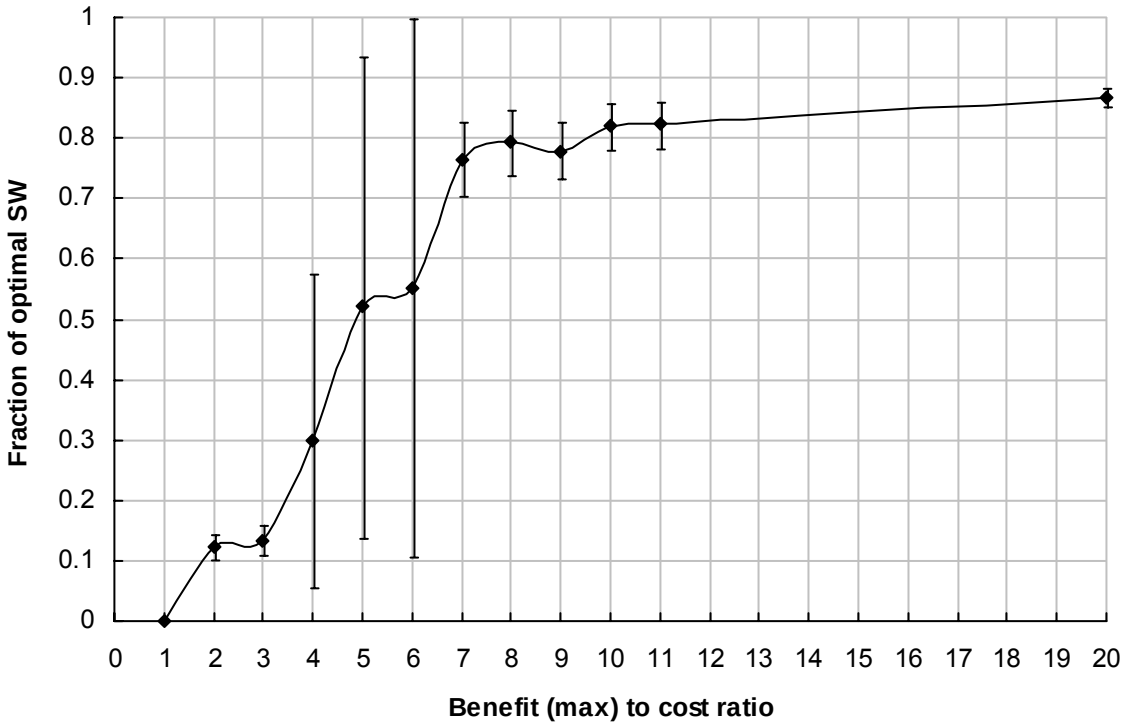

Figure 6.23 The success of a P2PWNC community also depends on the value of the service as expressed by the benefit-to-cost ratio. For very small ratios, the cooperators (ALLC, RECI) do not obtain ratings that are much higher than the under-providers (RAND) or the free-riders (ALLD). A transition occurs for ratios between 4 and 6. The confidence intervals indicate that the community may or may not succeed in evolving cooperation. This is not a problem for ratios higher than 6.

6.4 Summary of results

In the experiments presented in this chapter, we examined the behavior of the reciprocity algorithm, the gossiping algorithm, and the bootstrap algorithm. Peers who followed the three algorithms according to specification were said to follow the RECI strategy. We saw that when peers follow the RECI strategy, they contribute to peers who contribute to the community and do not contribute to free-riders. In addition, when confronting under-providers such as followers of the RAND strategy who only contribute with a probability, the RECIs identified and gave less benefit to the RANDs, thereby providing them with the incentive to switch to another strategy.

The price of this discriminating behavior is that RECIs do not always give the highest amount of benefit to other RECIs, and, therefore, the social welfare attained in the community is not the optimal. The optimal social welfare is the social welfare that would have been attained if all the peers in the community cooperated to the full after every request for service. However, we also saw that such an indiscriminately altruistic mixture is unstable. It is susceptible to invasions by free-riders; and once such an invasion begins, it is bound to lead to the total collapse of cooperation (see Experiment B-3).

Moreover, RECIs have a difficulty identifying other RECIs as the amount of available information decreases (see Experiment A-1). When the gossiping algorithm does not provide enough information on the history of the community, then, even in a strategy mixture that contains only RECIs, social welfare will practically stop increasing after a point (see Figure 6.3, Experiment A-1).

We saw that the patience parameter of the bootstrap algorithm is not crucial to the level of cooperation attained and that, in the long run, a small amount of patience is no different from a large amount of patience. The same applies to the community growth rate. P2PWNC communities that grow in membership quickly will do just as well in the long run as P2PWNC communities that grow in membership slowly.

We also saw that under the perfect matching mobility model, a peer that follows the attack we described in Section 5.1.7 and attempts to erase his outstanding debt in order to get the maximum amount of benefit is detected and punished by being offered service of lower quality (that is, he receives less benefit). We examined the same attack under the random waypoint mobility model, and the RECI strategy still detected and punished the "erase debt" strategy. Thus, in a community with RECI followers, the "erase debt" strategy appears disadvantaged compared to our algorithms.

In evolutionary settings, we saw that the RECI strategy could both invade a population, and withstand invasions by other strategies. We saw this behavior for a range of parameter values, and a range of simulation durations spanning two orders of magnitude.

For the specific benefit-cost model that we presented, it is in the interest of peers to follow the algorithms that we presented instead of an altruistic strategy (ALLC), a free-riding strategy (ALLD), or a strategy that contributes with a probability without examining community history (RAND). Although random drift can cause ALLC and RECI mixtures with ALLC percentages very near the RECI percentages, ALLCs still perform less well on average because of the constant invasion attempts by ALLDs. Meanwhile, RECIs "police" the community, and allow social welfare to continue to increase at a sustained rate.

Finally, the reciprocity algorithm does not rely on a central server that stores the entire community history, but on the decentralized storage subsystem, which gives only a partial view of this history. Nevertheless, the reciprocity algorithm still performs better than the alternatives.

Chapter 7

P2PWNC Protocol and Implementation

This chapter presents the P2PWNC protocol and performance measurements from our first prototype implementation of the protocol. The P2PWNC protocol has evolved over the course of this work. The protocol specification we present here is the third version (P2PWNC/3.0).

More information on the P2PWNC prototypes and the related proof-of-concept application (secure voice-over-IP telephony over P2PWNC for Windows Mobile smartphone clients) can be found on the P2PWNC project website [56], where the relevant demonstration papers [16, 55] are also available.

On the project website, the open-source first prototype implementation is also available for download. The first prototype includes firmware for the Linksys WRT54GS WLAN access point, configuration and management utilities, as well as P2PWNC clients written in C and Java. Program sources are ready to compile and install.

7.1 The P2PWNC protocol

7.1.1 Overview

Communication between the entities of a P2PWNC system is carried out through a set of seven messages. These seven messages are enough to specify all the necessary communication exchanges between the P2PWNC entities. These exchanges happen (1) between the P2PWNC AP and the P2PWNC client, (2) between the P2PWNC AP and the P2PWNC team server, and (3) between the P2PWNC client and the P2PWNC team server. In what follows, this set of protocol messages, their semantics, and their usage is described in detail.

The messages of the P2PWNC protocol are plain text. The main reason for this choice is that plain text messages are human-readable, which makes it more convenient when P2PWNC implementers debug their applications. This is the trend that several Internet Engineering Task Force (IETF) [86] protocols follow, such as the Session Initiation Protocol (SIP) [87]; SIP's format was the main inspiration behind the design of the P2PWNC protocol.

Each P2PWNC message contains in its first line a four-character message *type* and the *protocol version*. Then, one or more headers follow. A mandatory header is the

"Content-length" header; this indicates the number of octets in the *body* of the message, which is optional (in which case the Content-length header has a value of zero). All headers, as well as the first line, end with a *carriage return-line feed* (CRLF) character sequence. Binary data (such as public keys and digital signatures) are encoded using Base64 [88] encoding. Where present, message timestamps represent *Coordinated Universal Time* (UTC), and are encoded according to RFC 3339 [89].

The protocol runs over TCP/IP. Its messages are separated into two groups. Three messages (CONN, CACK, RREQ) are *inter-team*, in the sense that they are only exchanged between the P2PWNC client and the P2PWNC AP of two different teams. Three messages (QUER, QRSP, UPDT) are *intra-team*, in the sense that they are only exchanged between the P2PWNC AP and P2PWNC team server of the same team (QUER, QRSP), and the P2PWNC client and P2PWNC team server (UPDT) of the same team. The final message (RCPT) is used in both intra-team and inter-team communications. Note that all inter-team messages are always exchanged over the single WLAN link that connects a P2PWNC client and a P2PWNC AP.

The protocol also specifies that, for the inter-team messages, the P2PWNC client always use TCP port 9999, IP address 192.168.0.1 to find the P2PWNC AP module of the foreign team. This is possible because every time a P2PWNC client associates at the link-layer with a P2PWNC AP, it obtains an IP address over DHCP in the private range 192.168.0.x. The P2PWNC AP module is always available on the same address as the default IP gateway for that private IP subnet, and that address is 192.168.0.1.

When a P2PWNC client first associates at the link layer, the DHCP-allocated IP address starts off being blocked by the firewall co-located with the AP. The only communication possible is with the P2PWNC AP module on the aforementioned IP address and TCP port. If, later, client authentication succeeds and the P2PWNC AP is willing to cooperate, the firewall will allow packets from the IP address of the client to pass, and a NAT module will start forwarding packets toward the backbone, replacing the source address in the 192.168.0.x range with an appropriate routable IP address. The NAT module maintains the state that is necessary to pass incoming packets back to the appropriate P2PWNC clients using standard NAT techniques.

7.1.2 P2PWNC protocol messages

The messages of the P2PWNC protocol and short description of each one follows:

CONN (CONNection request) This message is sent by the P2PWNC client to request service from the P2PWNC AP that it currently visits. The client recognizes a P2PWNC access point because of the SSID link-layer beacons they transmit, which contain the prefix "P2PWNC-". CONN contains the member certificate that corresponds to the specific P2PWNC client. In this way, the P2PWNC AP learns the public key of the team that is requesting service. It can also verify that the specific user is member of that team by verifying the member certificate using the public key of the team found on the certificate. Although an attacker could perform a replay attack and present a certificate that is not his own, this attack will be detected as soon as the P2PWNC AP requests the first signed receipt. Since the attacker will not possess the private key that corresponds to the member certificate, he will not be able to sign the corresponding receipt (if he does try to sign, the receipt will not pass verification by the AP), and his connection will be terminated.

CACK (Connection request ACKnowledgment) This message indicates to the P2PWNC client that the P2PWNC AP has authenticated him and is willing to cooperate with him by contributing service. CACK comes as a response to the CONN request. The CACK response contains the timestamp that characterizes this WLAN session, and this timestamp will be used throughout the receipt generation protocol that will follow. All subsequent receipts that the P2PWNC client will sign and send to the P2PWNC AP during the WLAN session *must* include the same timestamp (but will have increasing *weights*).

Note that there is no concept of a negative CACK. If the P2PWNC AP is not willing to cooperate, it will simply terminate the underlying TCP/IP connection by sending a TCP FIN message. The overall philosophy of the P2PWNC protocol when it comes to inter-team messages is that teams are selfish and exchange the absolute minimum of messages with other teams at the P2PWNC layer.

RREQ (Receipt REQuest) Periodically, the P2PWNC AP requests from the P2PWNC client to acknowledge the specific amount of service that he has consumed during the session so far. This is equal to the total number of bytes the WLAN AP relayed for the WLAN client. This request is realized via the RREQ message. This message includes the amount of bytes relayed as the P2PWNC AP measures them. The first RREQ also contains, for the first time in the P2PWNC session, the contributing team's public key. The P2PWNC

client must reply with an RCPT message (within a reasonable time interval that is unspecified by the protocol).

RCPT (ReCeiPT) The RCPT message is the basic unit of information about P2PWNC resource contribution and consumption. The RCPT message contains (see also Section 4.6.1) the member certificate of the P2PWNC client (which contains the consuming team's public key, the consuming member's public key, and the signature of the consuming team that binds the specific consumer to the specific team); the contributing team's public key; the amount of traffic relayed by the WLAN AP so far (the receipt *weight*); and the session's timestamp. All the above are digitally signed by the P2PWNC client using the private key that corresponds to the member certificate presented in the initial CONN message. This signature is included in the RCPT message. By sending this message, the P2PWNC client acknowledges this amount of resource consumption. According to the receipt generation protocol (see Section 4.6.2), more than one receipt will probably be generated during the same WLAN session. These receipts will have increasing *weight* values and they will represent the P2PWNC client's responses to the periodical RREQ requests sent by the P2PWNC AP (the interval between two RREQs is left unspecified).

Eventually, only the last receipt of a session will be stored in the team server repository by the P2PWNC AP. Note that a P2PWNC AP cannot take advantage of more than one receipt from such a session in order to show additional contribution because these two receipts would have the same ID, and they would be recognized as the same by the other teams. A receipt's ID is the combination of the consuming member's certificate, the public key of the contributing team, and the timestamp fields—which are unique for each receipt in a P2PWNC system. (This is not strictly true as their uniqueness depends on the probability of key collisions during the generation of IDs by the teams—which is negligible. In addition, because the member certificate describes a specific individual, this individual is not able to initiate more than one P2PWNC session with the same contributing team at the same time.)

QUER (QUERy) This message is the first intra-team message that we present. It is used by the P2PWNC AP to communicate with its P2PWNC team server in order to ask whether and how much service to contribute to the requesting member, that is, the computation of the requesting member's Subjective Reputation Metric (SRM—see Section 5.1). It contains the public key of the team to which the requesting member belongs. (The P2PWNC team

server can be co-located with one of the team's APs in order to minimize the extra devices that a team's members will need to support, but this is an implementation detail, transparent to the P2PWNC protocol.)

QRSP (Query ReSPonse) This message is the second intra-team message that we present. A QRSP message is the reply to QUER. It is sent by the P2PWNC team server to the P2PWNC server and it contains the Subjective Reputation Metric (precise to two decimals) that was computed by the P2PWNC team server using its view of the receipt graph. It is up to the P2PWNC AP to make the final decision on how much service to contribute to the requesting member.

UPDT (UPDaTe) This message is the third and final intra-team message that we present. An UPDT message is sent by the P2PWNC client of a member to the P2PWNC team server whenever the P2PWNC client wants to refresh its client repository of receipts with fresh receipts from the team server. The message contains a timestamp. The team server that receives it will respond with a series of receipts whose timestamp is more recent than the one specified by the P2PWNC client, using RCPT messages. These receipts will not include receipts that show the consumptions of this team, as we already discussed (see Section 5.2—the update phase of the gossiping algorithm).

7.1.3 P2PWNC message examples

In this section, we give some examples of P2PWNC messages to show their syntax. The P2PWNC protocol supports both the RSA [90, 91] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [92, 93]. RSA cryptosystem parameters include the bit length of the keys and public exponent value (for us, fixed to 65537). For ECDSA, we used verifiably random curves over the F_p finite field. More specifically, we used the *sec160r1*, *secp192r1*, *secp224r1*, and *secp256r1* named curves [94, 95] for key lengths of 160, 192, 224, and 256 bits, respectively.

The protocol messages that contain cryptographic data include an "Algorithm" header, which indicates the signature algorithm used, and the length of the key.

CONN message The Base64-encoded data in the case of the CONN message contain the P2PWNC client's member certificate.

```
CONN P2PWNC/3.0
Content-length: 164
Algorithm: ECC160
BNibmxStfJlod/LnZubH6pzWHQqKyZFcSMjnZurmTe4KjCRkllhV93MEegPvCsxz
2oe/hqevoPSrwO1JLO/36J8HTIeyeKQqTCfx+EPxweAvYC/ZFb8URLa2faIbvSgD
3lm6Wa1S4cYlSWeSNmFzS/ebDFfzakqNSEs=
```

CACK message

```
CACK P2PWNC/3.0
Content-length: 0
Timestamp: Tue, 16 May 2006 17:26:41 +0000
```

RREQ message

```
RREQ P2PWNC/3.0
Content-length: 56
Algorithm: ECC160
Weight: 6336
BEXn8BHHViQ/YMyF2ny+KaI4YXz+W60uED7R8wZefDznyncfQKggzAc=
```

RCPT message The Base64-encoded data in the case of RCPT represent the fields of a receipt that are not human-readable, namely the consuming member's certificate, the contributor's public key, and the digital signature of the receipt. More details on the exact representation are given in the specification at [56]. Note that the digital signature also signs the weight and timestamp fields; to do that, the weight and timestamp are encoded in a compact representation that uses four bytes for each one of the two fields.

```
RCPT P2PWNC/3.0
Content-length: 272
Algorithm: ECC160
Timestamp: Tue, 16 May 2006 17:26:41 +0000
Weight: 6336
BNibmxStfJlod/LnZubH6pzWHQqKyZFcSMjnZurmTe4KjCRkllhV93MEegPvCsxz
2oe/hqevoPSrwO1JLO/36J8HTIeyeKQqTCfx+EPxweAvYC/ZFb8URLa2faIbvSgD
3lm6Wa1S4cYlSWeSNmFzS/ebDFfzakqNSEsERefwEcdWJD9gzIXafL4pojhhfP5b
rS4QPtHzBl58POfKdx9AqCDMBxRoGALKJSJYYXlsrwtiyZJKvPlU5B3lWrFuL25P
d+kv2iMVRElXk/4=
```

QUER message

```
QUER P2PWNC/3.0
Content-length: 56
Algorithm: ECC160
BNibmxStfJlod/LnZubH6pzWHQqKyZFcSMjnZurmTe4KjCRkllhV93M=
```

QRSP message example

```
QRSP P2PWNC/3.0
Content-length: 0
Subjective-reputation: 0.90
```

UPDT message example

```
UPDT P2PWNC/3.0
Content-length: 0
Timestamp: Sat, 13 May 2006 17:26:41 +0000
```

7.2 Implementation and measurements

We implemented the first version of the P2PWNC protocol (P2PWNC/1.0) on the Linux-based Linksys WRT54GS WLAN AP [30]. This AP is currently (2006) retailing for less than $50. In this P2PWNC implementation, in addition to the P2PWNC AP, the P2PWNC team server can also be co-located with the Linksys AP. In theory, one of the P2PWNC APs of a team can be chosen to serve as team server also. Our software modules were included on the AP's firmware. We also implemented the client side of the protocol as a Java application that runs on Windows and Linux, and as a C application that runs on Linux. See the project website [56] for more details regarding the latest version of the P2PWNC AP module and the P2PWNC client for the QTEK smart-phone [13].

We conducted experiments to test the performance of the software running on the Linksys AP. For comparison, we also ran the software on an AMD Athlon XP 2800 laptop. Table 7.1 shows the specifications of these two platforms.

Table 7.1 Platform specifications for the experiments of Section 7.2

Characteristic	Athlon XP 2800	Linksys WRT54GS
CPU speed	2.08 GHz	200 MHz
CPU type	AMD Athlon XP 2800	Broadcom MIPS32
RAM	512 MB	32 MB
Storage	60 GB HD	8 MB Flash, 32 KB NVRAM
Operating system	Linux kernel 2.4.18 (Red Hat Linux 8.0)	Linux kernel 2.4.18 (Broadcom specific)
Cryptographic library	OpenSSL 0.9.8 beta 5	OpenSSL 0.9.8 beta 5
Compiler	gcc 3.2	gcc 3.2
Compiler optimizations	-O3	-O3 –mcpu=r4600 –mips2

The P2PWNC AP uses the Linux *iptables* firewall for network access control. We built our own Linux kernel module for measuring the volume of relayed traffic per P2PWNC client. When P2PWNC clients associate with the AP, they are assigned dynamic IP addresses from an address pool via DHCP, and denied Internet access until the P2PWNC server receives a QRSP from the team server. When a session ends (RREQ timeout) the particular IP address is once again blocked, and measurement of traffic for or from that address is stopped.

We did tests to study the performance of both the RSA and the ECDSA signature schemes. The operations of interest were the signing and the verification of digital signatures. In our protocol, P2PWNC clients sign receipts and WLAN APs verify them. Tables 7.2 and 7.3 show the results of these tests, measured in milliseconds. The tables show the pure CPU time spent on executing the corresponding operation (measured with the Linux *times* function), averaged after 10 runs. Of these tables, the columns of interest are the ones that contain the Linksys verification times and the PC signature times, as these are the actions that will be taken by this type of devices according to our protocol. The other columns are included for completeness.

ECDSA is faster for signatures than for verifications. In addition, the use of ECDSA results in smaller receipts because ECDSA keys and signatures are shorter than their RSA counterparts for the same security level. The smallest receipt in P2PWNC, when all keys and signatures use 160-bit Elliptic Curve Cryptography (ECC), requires 211 bytes (3 ECC public keys, represented without point compression, requiring 41 bytes each, two ECDSA signatures requiring 40 bytes each, and two 4-byte integers representing the timestamp and weight fields). With 1024-RSA keys, the receipt size is 648 bytes. For both sizes of receipts, modern cell phones [13] can theoretically store thousand of receipts in their mobile repositories before they run out of memory.

Table 7.2 Cryptographic operation performance: Signing

	Athlon XP 2800		Linksys WRT54GS	
Bit length (RSA/ECC)	RSA (ms)	ECC (ms)	RSA (ms)	ECC (ms)
1024/160	9.0	1.3	300.6	20.3
1536/192	25.9	1.2	655.6	18.5
2048/224	47.3	1.4	1529.0	23.4
3072/256	149.1	1.7	3939.0	73.1

Table 7.3 Cryptographic operation performance: Verification

	Athlon XP 2800		Linksys WRT54GS	
Bit length (RSA/ECC)	RSA (ms)	ECC (ms)	RSA (ms)	ECC (ms)
1024/160	0.4	6.5	12.3	114.7
1536/192	0.8	6.0	21.4	99.9
2048/224	1.3	7.1	37.9	135.7
3072/256	2.8	8.6	75.3	453.0

Note that to verify an RSA signature is faster than signing one, while the opposite is true for ECDSA signatures. Note also how the ECC-192 scheme (the *secp192r1* named curve) is faster than the ECC-160 scheme for both signature verification and signature generation.

Therefore, the two schemes represent a tradeoff: With an ECC scheme, more receipts can be stored in the P2PWNC client's mobile repository. On the other hand, the Linksys WRT54GS needs 114.7 ms to verify an ECC-160 signature but only 12.3 ms to verify an RSA-1024 signature (see Table 7.3). When the merging phase of the gossiping algorithm runs, the P2PWNC AP will need to verify several receipts, so short verification time is important if the team server is located on the P2PWNC AP.

We implemented the team server with the team receipt repository to run on the Linksys AP itself. The receipt repository needs to support dictionary and graph operations. It should support efficient receipt insertion, deletion, search, and maximum flow (maxflow) computation. A composite data structure was built to accommodate this. The structure includes hash tables that store pointers to receipts and team IDs in order to achieve fast look-ups. A red-black tree [96] structure keeps receipts sorted by their timestamp. Each tree node corresponds to a receipt. A red-black tree supports logarithmic-time insertion, deletion, and searching for receipts according to their timestamp. Finally, there is an adjacency-list representation of the receipt graph, with each red-black tree node also storing a pointer to the respective graph edge. A FIFO variant of the *push-relabel* maxflow algorithm [97] has been implemented. Its $O(V^3)$ worst case running time is long; therefore, we used the global relabeling heuristic [97, 98], which yielded dramatic performance improvements.

We measured the performance of the FIFO-based push-relabel algorithm with the global relabeling heuristic for various random graph instances. In our experiments, we

created random directed graphs comprising 1000 and 10,000 receipts, and 100 and 1000 teams. Table 7.4 shows the pure CPU time spent on executing the algorithm (measured with the Linux *times* function). Each reported value is the average time spent on the execution of the maxflow algorithm for 20 random source-destination pairs of the same graph. Time is measured in milliseconds.

Table 7.4 Maxflow algorithm performance

	Athlon XP 2800		Linksys WRT54GS	
Number of receipts	100 teams (ms)	1000 teams (ms)	100 teams (ms)	1000 teams (ms)
1000	0.43	0.23	12.64	3.75
10,000	5.88	12.72	59.27	134.04

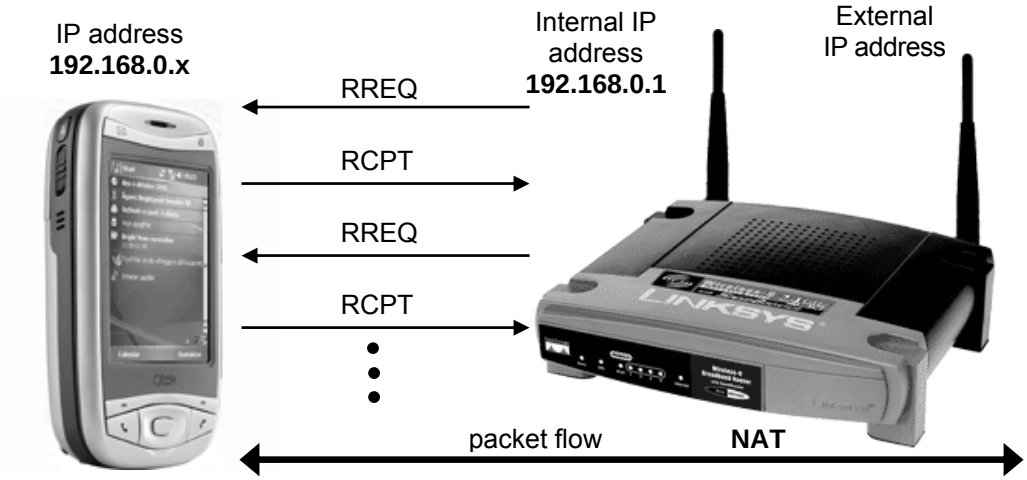

Figure 7.1 The QTEK 9100 WLAN-enabled cell phone and the Linksys WRT54GS AP. P2PWNC software for both platforms is available. The figure shows the addressing scheme supported by our architecture. The P2PWNC AP acts as NAT (Network Address Translator) router, relaying the client's IP packets to the backbone, and supporting multiple clients at the same time while using only one external IP address (which could be a public Internet address, or another type, depending on the backbone).

7.3 Summary

The P2PWNC protocol is designed as a lightweight vendor-neutral protocol that can be implemented on top of resource-constrained devices. Figure 7.1 shows two such devices. Our implementation of the P2PWNC protocol has shown that the P2PWNC AP

and team server functionality required by the scheme can be implemented on top of a $50 WLAN AP. We claim that most of the implementation ideas presented here can be carried over to other similar platforms. Moreover, the P2PWNC protocol itself is simple (7 messages only) and easily extensible, as our ongoing work [56] shows.

Chapter 8

Discussion and Future Work

This chapter discusses several aspects of the P2PWNC scheme that were hinted at in previous chapters but were not part of the dissertation's main argument. These include: (1) problems relating to real-world P2PWNC deployments, (2) problems relating to the P2PWNC system models and possible extensions of these models, (3) algorithmic improvements, (4) variants and extensions to the P2PWNC evaluation framework, and (5) additional implementation options.

Sharing-friendly ISPs For a P2P WLAN sharing scheme like P2PWNC to function, the contributors need to relay traffic for other P2PWNC peers through their Internet connections. In case of a residential Internet connection that goes through a commercial Internet Service Provider, it is possible that the ISP's *Acceptable Use Policy* (AUP) does not allow sharing of the Internet connection with people outside the subscribing household. This possibility was already considered in Section 6.1.3 as one potential cost generator for P2PWNC contributors.

There are several ISPs that dictate sharing-friendly AUPs [99]. However, larger telecom organizations that offer Internet access are not currently sharing-friendly. In any case, the use of NAT in the subscribing household (and on the P2PWNC AP) multiplexes many IP flows over a single connection, and it is not straightforward for the ISP to detect which ones originated within the household. However, overcautious subscribers may be unwilling to violate the AUP in principle; and ISPs need only *suspect* AUP violation, by examining 24-hour usage patterns for example. Depending on the contract, the ISP can then discontinue service or offer service of lower quality as a disincentive. It is possible that, in the future, a convincing business case will be made *for* sharing-friendly ISPs. Meanwhile, options other than commercial ISPs for backhaul connectivity are available (see below).

P2PWNC and Wireless Community Networks In order to circumvent commercial ISPs completely, a P2PWNC system can be used on top of a *Wireless Community Network* (WCN—see Section 1.3.2). In such a network, a P2PWNC AP will also be a node in the WCN, and traffic will flow from WCN node to WCN node, bypassing wired ISPs. If no

WCN node connects to the Internet, this limits the usefulness of P2PWNC. If, however, the primary application running on top of P2PWNC is video- and voice-over-IP using WLAN-enabled cell phones, and if calls mainly originate and terminate within the same urban area (town, city, metropolis), then a WCN covering this area could still function as an alternative to commercial cellular networks in the same area.

In (non-mesh) WCNs, the nodes establish point-to-point wireless links with each other, which requires human intervention. This guarantees that two neighboring WCN nodes generally have a long-lasting relationship that is mutually beneficial. The two nodes at the end of a link know and recognize each other because the node owners needed to communicate and cooperate in order to set up the particular link in the first place. Because this applies to all WCN links, cooperation extends to the whole network. In practice, WCN backbones that use point-to-point links appear to work [32, 33].

It is another matter entirely, however, for a WCN node to allow a random mobile client to access the AP/node. This is where P2PWNC becomes useful. P2PWNC can effectively detect and guarantee that the mobile user himself also owns a node in the WCN and, more importantly, that he actually forwards traffic for other clients through his combination P2PWNC AP/WCN node. P2PWNC is thus used as an incentive scheme to stimulate participation in WCNs. This way, the WCN's normally open and self-organized nature is not compromised, the WCN is not exposed to free-riding mobile clients, and, at the same time, the WCN can offer the equivalent of a citywide cellular service.

To respect a WCN's open and self-organized nature means not to require a central authority that must keep track of which nodes are part of the WCN. Note that because WCNs require an addressing scheme in order to function, an authority that distributes IP addresses within the WCN is required [33]. By adopting P2PWNC in the WCN, however, this authority does not require to take up more complex functions like, for example, issuing certified IDs to mobile clients and brokering transactions between a prospective contributor and a prospective consumer (in a way that is also robust to sophisticated free-riders).

Congestion It was assumed that congestion is not a problem in P2PWNC and the evaluation of P2PWNC did not include cases where more than one visitor request service from the same P2PWNC AP at the same time, or APs that cannot provide the amount of bandwidth that the consumer requires. In practice, congestion may be a problem in deployed P2PWNC systems. There are always two distinct links used by a P2PWNC AP:

the WLAN link, and the backhaul link, either of which can become congested. In general, it is not easy to say which one of the two links will be the bottleneck. In a P2PWNC system that uses DSL/Cable connections for backhaul connectivity, the likely bottleneck is this wired link; but the unpredictability of the wireless channel coupled with the fact that modern DSL connections have capacity of 8 Mbps [100] makes the wireless link a likely bottleneck also. In a WCN-based P2PWNC (see above), if the backbone point-to-point links use IEEE 802.11a at 54 Mbps and the APs use 802.11b at 11 Mbps (a common arrangement [33]), then the 802.11b link is the likely bottleneck. In all these cases, standard techniques for allocating specific amounts of bandwidth to specific connections—at the P2PWNC AP/router level—can protect the owner of the P2PWNC AP/router from over-consuming P2PWNC clients. In addition, the amount of bandwidth offered to a P2PWNC client can depend on the Subjective Reputation Metric of that client. The prototype of P2PWNC/3.0 [55, 56] that we hinted at in the previous chapter already offers this rudimentary quality-of-service functionality.

Extended benefit-cost model for P2PWNC A benefit-cost model for P2PWNC that acknowledges that, often, people simply enjoy sharing their own resources with others (effectively acting altruistically) may be more realistic compared to the current model of a constant (unit) positive cost, and zero benefit, per P2PWNC contribution. In addition, a benefit-cost model that takes into account the current congestion level when evaluating cost may also be realistic. In general, one must experiment with several benefit-cost models, with non-uniform benefit-cost functions, and with benefit-cost functions that depend on the particular WLAN application used by P2PWNC users in order to strengthen the result that a P2PWNC system can generate positive social welfare in general. In addition, a possible spirit of competition among P2PWNC teams, similar perhaps to the one observed among SETI@home teams [46], could increase both individual enjoyment and overall P2PWNC cooperation levels, and should be factored into the extended benefit-cost model.

Extended simulator model The current simulator model for P2PWNC makes several simplifying assumptions. It represents P2PWNC receipts using unit weights, it simulates mobility using three basic mobility models, and it simulates evolution through one specific learning model. In addition, the universe of available strategies is composed of three strategies that do not examine the receipt graph (ALLC, ALLD, and RAND), the proposed

P2PWNC reciprocity algorithm (RECI), and two selfish/adversarial strategies ("erase debt" and "over-consumption"—see Experiments A-7, A-8 in Section 6.2). Possible extensions to the simulator include (1) the use of non-unit receipts, (2) the use of additional mobility models based on real-world traces of cellular subscriber activity (to be used as a good approximation of a P2PWNC community that mainly places and receives video- and voice-over-IP calls), and (3) additional selfish/adversarial strategies that attack the system, for example, during peer bootstrap, or by manipulating the gossiping algorithm in their favor.

Team formation and management The problem of how P2PWNC teams are formed and managed needs further investigation. In addition, teams may not be the ideal peering entities in P2PWNC. Teams solve two main problems. First is the real-world issue of households with many members and a single WLAN AP; it would be unreasonable to require that these households install an AP for each member, so grouping the household members in the same team makes sense. However, another approach, without using teams, would be to have one AP rotate its ID periodically so that it effectively "belongs" to a different household member during each time slot.

Peripheral peers The concept of teams helps to solve the problem of individuals that wish to participate in P2PWNC but reside in an area where there are simply not enough users to serve. These individuals can team with others that are willing to accept them in their team, so that the team as a whole can still bring its consumption/contribution ratio near one-to-one.

Another way to solve the problem of peers that reside in the periphery of a P2PWNC system, such as the outskirts of a city, would be to increase the weight of the receipts they request by a factor that depends on the remoteness of their location. Normally, P2PWNC clients would refuse to sign receipts showing an amount of consumption different from the one they measure. This was, after all, one of the foundations of the receipt generation protocol: The contributor needs the receipts, and the consumer needs the service; splitting the session into smaller parts that are acknowledged by intermediate receipts guarantees that neither the contributor nor the consumer ever has a significant advantage.

These dynamics would change if contributors were free to set arbitrary weights. A contributor with a location advantage, that is, a contributor that is the sole P2PWNC contributor covering a specific area, could use a multiplicative factor when measuring the

weight. The consumer could still sign depending on his valuation of the service (these receipts would increase the consuming team's apparent consumption by a number that did not correspond to actual consumption). Such receipts with larger weights could, however, make up for the fact that the specific contributor does not receive consumption requests often.

Global P2PWNC The P2PWNC scheme focuses on urban areas, and the mobility models adopted in the P2PWNC simulator assume citywide scales where a few thousand P2PWNC teams interact. A more general mobility model would consider a global P2PWNC system, and include mobility patterns appropriate for global travelers, in an attempt to predict whether global WLAN roaming through P2PWNC is viable.

The problem is one of information: the receipt repositories on team servers depend on the number of teams/peers (more specifically, they increase linearly with the number of peers). One possible solution would be to use *super-peers* in the following manner. One super-peer could be the "representative" of each different P2PWNC "area". If the super-peers were well-known globally, a prospective contributor that could not detect indirect or direct debt to a prospective consumer would ask the super-peer from the consumer's area whether *he* would offer service to the specific consumer if the specific consumer had been visiting the super-peer at the time. If the prospective contributor could trust the super-peer to give an honest answer, then he could cooperate with the prospective consumer depending on that answer. There are three problems, however. First, there is the problem of how to appoint super-peers and distribute lists of super-peers globally. Second, how to trust that their answers are honest. Third, how to encourage prospective contributors to follow the suggestion of the super-peer when, with high probability, the receipts that he will receive from the roaming consumer would not be useful: The receipts received through gossiping would include receipts from the consumer's home area, and the new receipt issued would connect the contributor with a receipt tree in that area (see also relevant discussion in Section 6.2.5), which would not be useful if the contributor mainly consumes in *his own* home area, and not the home area of the consumer.

Denial-of-Service attacks There are several types of Denial-of-Service (DoS) attacks to which P2PWNC is susceptible. A basic attack is wireless interference at the physical layer, which could stop peers from being able to consume or contribute. Such malicious irrational

attacks would not be beneficial to the attackers. More interestingly from a selfish perspective, an attacking prospective contributor could attack neighboring prospective contributors by directing jamming signals toward their APs so that the mobile clients in the area detect only his own P2PWNC AP and request service from him alone. Other malicious attacks could involve IP-layer DoS attacks against team servers. By making team servers inaccessible to the APs of a team, the APs would not be able to consult the reciprocity algorithm; if they denied consumption requests as a result, the consumers could then try to use nearby APs—belonging perhaps to the attacker. However, the effect of this attack is minimized because the P2PWNC protocol never reveals the IP address of team servers to other teams.

Another DoS attack would involve the gossiping protocol: A consumer could overwhelm a prospective contributor with numerous fake receipts. Although these receipts would not improve the consumer's Subjective Reputation Metric (because of the use of the maxflow algorithm to detect direct and indirect debt—see also relevant discussion on Section 2.4 and Section 5.1.6), they could harm the contributing peer and the system as a whole by polluting repositories, which are finite in size, with useless information, bringing cooperation levels down.

The effect of all the attacks above is limited to specific peers or to a small number of peers. Because the attacks do not bring added benefit to the attacker (with the possible exception of the first attack), they are irrational actions against which only simple countermeasures may be required. In the case of the directed jamming attack, it is relatively straightforward to physically detect the attacker—so punishment could be possible.

Implementation extensions There are two interesting implementation extensions of practical importance. The first is the issue of supporting fast handovers among P2PWNC APs belonging to different teams in order to support mobile users that place and receive video- and voice-over-IP calls. WLAN standards for fast handovers do not yet exist, but market demand guarantees that they will appear. In addition, faster variants of the Dynamic Host Configuration Protocol (DHCP) will eventually appear (this would be needed because P2PWNC clients obtain a new IP address each time they change P2PWNC AP). The final step, therefore, in order to support fast handovers would be to ensure that the P2PWNC CONN-QUER-QRSP-CACK protocol sequence, and the associated execution of maxflow algorithms by the team server, is not the bottleneck in the handover procedure. (In all cases,

it is assumed that the client application can recover gracefully from the possible loss of IP packets during the handover.)

A second possible extension that could affect the aforementioned one includes decentralizing the P2PWNC team server among team members. One simple approach would be for all team APs to store replicas of the team server repository, and act as team servers themselves. The problem then becomes one of synchronization of information among team APs. However, this also guarantees that the potentially expensive communication between P2PWNC AP and P2PWNC team server will not represent the bottleneck in the handover procedure mentioned above. Another decentralized option, albeit slower, would be for a team's P2PWNC APs to be organized in a structured overlay like a Distributed Hash Table (see Section 1.4) in which the receipts are stored.

Algorithmic improvements Two specific improvements to the P2PWNC algorithms include dynamic discovery of optimal parameter values for the mobile repository size, the team server repository size, and the patience parameter. In addition, an algorithm that merges only the "most valuable" receipts, and can therefore keep the size of the mobile/client repository small, could be valuable.

Collusion In the current model, P2PWNC peers (teams) never communicate with each other; the only way for them to interact is when a P2PWNC client from one team requests service from the P2PWNC AP of another team. Scenarios where collusion among different teams is possible would represent an entirely new class of adversarial strategies that may need to be examined. The use of the maxflow algorithm ensures that naïve collusion (see Chapter 3—Definitions) would bring no benefit to the colluders. However, there are perhaps several scenarios with collusion using *front peers* (see Chapter 3 for a definition) that could be beneficial to all the colluders in a colluding group.

Tamperproof modules and the use of authorities A final approach to designing a completely different P2P WLAN sharing scheme would be to assume tamperproof modules or the existence of authorities. This would represent a fundamental change that could lead to the creation of a completely different system architecture.

However, both assumptions are strong. The first, because it only takes one sophisticated hacker to make hacked versions available to all users (as the experience with Kazaa-

Hack has shown [43]); and the question then becomes how fast the word on this hacked version spreads. The second assumption (relying on authorities) appears unreasonable because the basic P2PWNC service (WLAN/Internet access) is straightforward to implement. Where one authority appears (such as FON [101]), a second one could also appear causing the system to fragment into uncooperative administrative domains. This would bring us back full circle to the original problem—uncooperative WLAN administrative domains—that motivated this work (see Section 1.1).

Chapter 9

Conclusion

The guiding vision behind the design of the Peer-to-Peer Wireless Network Confederation scheme and its algorithms was that of a broadband subscriber who purchases a P2PWNC-enabled WLAN AP, sets it up at home in "sharing" mode, and earns the right to roam within his city enjoying high-speed wireless access through other P2PWNC APs. We wanted the scheme to be: (1) open to all, with no registration procedure; (2) free to use, with reciprocity as the only driving force; and (3) simple to implement. We believe that our simulation results and our prototype implementation represent a first step towards proving that such a system could be viable in practice.

Support for free identities is a challenging aspect of the P2PWNC design. On the other hand, the freedom of not having to register with authorities could permit massive P2PWNC systems to grow organically.

Our simulations took a high-level view of the system. We showed how a reciprocal strategy could outperform other strategies, and, in the process, help sustain high cooperation levels. The proposed reciprocal strategy achieved this without having access to the complete history of the system. However, there was a tradeoff, and the community as a whole paid a constant price. Cooperators punished each other with reduced levels of cooperation. Their overly cautious approach was necessary: A strategy mixture where indiscriminate altruists prevail is unstable, practically inviting the invasions of free-riders.

On the implementation front, we showed that it is possible to implement the scheme using low-cost hardware.

There is opportunity for further work in this area. A new P2P WLAN sharing design could start by assuming a core group of users who know and trust each other. These super-peers, or *founders*, would be trusted to monitor a P2P WLAN system as it grows into maturity. In the process, they would set the standards for those who would follow. Such an assumption would give designers of future P2PWNCs additional freedom in their implementation choices. We believe that current Wireless Community Networks and their members represent an ideal test-bed for schemes like P2PWNC.

Bibliography

[1] M. Giesler and M. Pohlmann, "The anthropology of file sharing: consuming Napster as a gift," Association for Consumer Research Conference, Atlanta, GA, 2002.

[2] IEEE 802.11b-1999. Supplement to 802.11-1999 Wireless LAN MAC and PHY specifications: Higher Speed Physical Layer Extension in the 2.4 GHz band.

[3] Kazaa history. Available at http://en.wikipedia.org/wiki/Kazaa

[4] IEEE 802.11g-2003. Amendment 4 to 802.11-1999 Wireless LAN MAC and PHY specifications: Further Higher-Speed Physical Layer Extension in the 2.4 GHz band.

[5] IEEE 802.11e-2005. Amendment 8 to 802.11-1999 Wireless LAN MAC and PHY specifications: Medium Access Control Quality of Service Enhancements.

[6] IEEE 802.11i-2004. Amendment 6 to 802.11-1999 Wireless LAN MAC and PHY specifications: Medium Access Control Security Enhancements.

[7] A. Stubblefield, J. Ioannidis, and A. D. Rubin, "Using the Fluhrer, Mantin, and Shamir attack to break WEP," Network and Distributed Systems Security Symposium (NDSS'02), San Diego, CA, 2002.

[8] IEEE Std 802.11. Wireless LAN Medium Access Control and Physical Layer Specifications. ISO/IEC 8802-11:1999(E), 1999.

[9] E. C. Efstathiou and G. C. Polyzos, "A peer-to-peer approach to wireless LAN roaming," ACM Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH'03), San Diego, CA, 2003.

[10] IEEE Std 802.1X. Port Based Network Access Control.

[11] NoCatAuth Home Page. Available at http://nocat.net

[12] R. Flickenger, *Building Wireless Community Networks*, O'Reilly & Associates, 2001.

[13] QTEK 9100 GSM Pocket PC Phone Edition with WiFi and Windows Mobile 5.0. Available at http://www.qtek.gr/9100.html

[14] R. Horn and P. Demain, "Cellular and WLAN convergence," Wireless Broadband Forum, Cambridge, UK, 2002.

[15] E. C. Efstathiou, P. A. Frangoudis, and G. C. Polyzos, "Stimulating participation in wireless community networks," IEEE INFOCOM 2006, Barcelona, Spain, 2006.

[16] E. C. Efstathiou, F. A. Elianos, P. A. Frangoudis, V. P. Kemerlis, D. C. Paraskevaidis, G. C. Polyzos, and E. C. Stefanis, "The peer-to-peer wireless network confederation scheme," IEEE INFOCOM 2006 Demo Session, Barcelona, Spain, 2006.

[17] N. Negroponte, "Being wireless," *Wired 10.10*, 2002.

[18] D. Tweney, "FON hopes its hotspots will rival cellular." Available at http://dylan.tweney.com/2006/04/21/fon-aims-for-ubiquitous-wi-fi

[19] Google Inc. Home Page. Available at http://www.google.com

[20] R. Dawkins, *The Selfish Gene*, Oxford University Press, 1976.

[21] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, The MIT Press, 1994.

[22] R. Krishnan, M. D. Smith, Z. Tang, and R. Telang, "The virtual commons: why free-riding can be tolerated in file sharing networks," Available at SSRN: http://ssrn.com/abstract=450241 or DOI: 10.2139/ssrn.450241, November 2004.

[23] The Bluetooth Special Interest Group (SIG) Home Page. Available at http://www.bluetooth.com

[24] IEEE Std 802.11. Wireless LAN Medium Access Control and Physical Layer Specifications. ISO/IEC 8802-11:1997(E), 1997.

[25] Wi-Fi Alliance Home Page. Available at http://www.wi-fi.org

[26] N. Golmie, N. Chevrollier, and O. Rebala, "Bluetooth and WLAN coexistence: challenges and solutions," *IEEE Wireless Communications*, 10(6), Dec. 2003.

[27] R. Bruno, M. Conti, and E. Gregori, "Throughput analysis of UDP and TCP flows in IEEE 802.11b WLANs: a simple model and its validation," 2005 Workshop on Techniques, Methodologies, and Tools for Performance Evaluation of Complex Systems (FIRB-PERF'05), 2005.

[28] Proxim Corporation white paper, "Maximizing your 802.11g investment," 2003.

[29] Inter-Access Point Protocol (IAAP—IEEE 802.11F) description. Available at http://en.wikipedia.org/wiki/IEEE_802.11F

[30] Linksys WRT54G Firmware Source Code. Available at http://www.linksys.com

[31] OpenWrt Linux Distribution. Available at http://openwrt.org

[32] Seattle Wireless Home Page. Available at http://www.seattlewireless.net

[33] Athens Wireless Metropolitan Network Home Page. Available at http://www.awmn.gr

[34] The Cloud Home Page. Available at http://www.iwantwifihere.com

[35] NYCwireless Home Page. Available at http://www.nycwireless.net

[36] Bay Area Wireless Users Group Home Page. Available at http://www.bawug.org

[37] Wireless Philadelphia Project Home Page. Available at http://www.phila.gov/wireless

[38] Wired News: "Google offers San Francisco Wi-Fi—for free" Available at http://www.wired.com/news/wireless/0,1382,69059,00.html

[39] Napster Home Page. Available at http://www.napster.com

[40] Gnutella Home Page. Available at http://www.gnutella.com

[41] E. Adar and B. A. Huberman, "Free-riding on Gnutella,"*First Monday*, 5(10), 2000.

[42] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing system," Multimedia Computing and Networking 2002 (MMCN'02), San Jose, CA, 2002

[43] K-Hack (Kazaa Hack) Home Page. Available at http://www.khack.com

[44] BitTorrent Home Page. Available at http://www.bittorrent.com

[45] B. Cohen, "Incentives build robustness in BitTorrent," Peer-to-Peer Economics Workshop (p2pecon'03), Berkeley, CA, 2003.

[46] SETI@home Home Page. Available at http://setiathome.berkeley.edu

[47] Folding@home Home Page. Available at http://folding.stanford.edu

[48] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: enabling scalable virtual organizations," *International Journal of Supercomputer Applications,* 15(3), 2001.

[49] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, 11(1), 2003.

[50] A. Rowstron and P. Druschel, "Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems," IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, 2001.

[51] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, 22(1), 2004.

[52] P. Maymounkov and D. Mazieres, "Kademlia: a peer-to-peer information system based on the XOR metric," 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, 2002.

[53] S. Ratsanamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," ACM SIGCOMM, San Diego, CA, 2001.

[54] K. Aberer, P. Cudre-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, "P-Grid: a self-organizing structured P2P system," *SIGMOD Record*, 32(2), 2003.

[55] E. C. Efstathiou, F. A. Elianos, P. A. Frangoudis, V. P. Kemerlis, D. C. Paraskevaidis, G. C. Polyzos, and E. C. Stefanis, "Practical incentive techniques for wireless community networks," 4th International Conference on Mobile Systems, Applications, and Services (MobiSys 2006) Demo Session, Uppsala, Sweden, 2006.

[56] The Peer-to-Peer Wireless Network Confederation Project Home Page. Available at http://mm.aueb.gr/research/P2PWNC

[57] R. Axelrod and W. D. Hamilton, "The evolution of cooperation," *Science*, vol. 211, 1981.

[58] Y. Benkler, "Sharing nicely: on shareable goods and the emergence of sharing as a modality of economic production," *The Yale Law Journal*, vol. 114, 2004.

[59] T.-W. Ngan, D. S. Wallach, and P. Druschel, "Enforcing fair sharing of peer-to-peer resources," 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), Berkeley, CA, 2003.

[60] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "KARMA: a secure economics framework for P2P resource sharing," 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, 2003.

[61] L. Buttyan and J.-P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks,"*ACM/Kluwer Mobile Networks and Applications*, 8(5), 2003.

[62] N. Ben Salem, J.-P. Hubaux, and M. Jakobsson, "Reputation-based Wi-Fi deployment," *Mobile Computing and Communications Review* (MC2R), July 2005.

[63] K. G. Anagnostakis and M. B. Greenwald, "Exchanged-based incentive mechanisms for peer-to-peer file sharing," 24th International Conference on Distributed Computing Systems (ICDCS 2004), Tokyo, Japan, 2004.

[64] L. P. Cox and B. D. Noble, "Samsara: honor among thieves in peer-to-peer storage,"19th ACM Symposium on Operating System Principles (SOSP'03), Bolton Landing, NY, 2003.

[65] A. Blanc, Y.-K. Liu, and A. Vahdat, "Designing incentives for peer-to-peer routing," IEEE INFOCOM 2005, Miami, FL, 2005.

[66] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," 12th International World Wide Web Conference (WWW'03), Budapest, Hungary, 2003.

[67] S. Lee, R. Sherwood, B. Bhattacharjee, "Cooperative peer groups in NICE," IEEE INFOCOM 2003, San Francisco, CA, 2003.

[68] R. Axelrod and D. Dion, "The further evolution of cooperation," *Science*, vol. 242, 1988.

[69] J. W. Weibull, "What have we learned from evolutionary game theory so far?" *Scandinavian Working Papers in Economics* (S-WoPEc), no. 487, 1998.

[70] J. Douceur, "The Sybil attack," 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, 2002.

[71] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," ACM Conference on Electronic Commerce (EC'04), New York, NY, 2004.

[72] M. Feldman and J. Chuang, "The evolution of cooperation under cheap pseudonyms," 7th IEEE Conference on E-Commerce Technology (CEC), Munich, Germany, 2005.

[73] E. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *Journal of Economics and Management Strategy*, 10(2), 1998.

[74] S. Capkun, L. Buttyan, and J.-P. Hubaux, "Self-organized public key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, 2(1), 2003.

[75] Pretty Good Privacy (Phil Zimmermann's Home Page). Available at http://www.philzimmermann.com

[76] M. Felegyhazi, S. Capkun, and J.-P. Hubaux, "SOWER: a self-organized wireless network for messaging," ACM Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH'04), Philadelphia, PA, 2004.

[77] C. Courcoubetis and R. Weber, "Incentives for large P2P systems," *IEEE Journal on Selected Areas in Telecommunications*, 24(5), 2006.

[78] B. Yang and H. Garcia-Molina, "PPay: micropayments for peer-to-peer systems," 10th ACM Conference on Computer and Communications Security (CCS'03), Washington, DC, 2003.

[79] A. Odlyzko, "The case against micropayments," *Financial Cryptography 2003*, J. Camp and R. Wright, eds., *Lecture Notes in Computer Science*, Springer, 2003.

[80] É. Tardos and K. D. Wayne, "Simple generalized maximum flow algorithms," 6th International Conference on Integer Programming and Combinatorial Optimization, 1998.

[81] R. Hoffmann, "The ecology of cooperation," *Theory and Decision*, vol. 50, 2001.

[82] S. Bowles and H. Gintis, "The evolution of strong reciprocity: cooperation in heterogeneous populations," *Theoretical Population Biology*, vol. 65, 2004.

[83] The 3rd Generation Partnership Project (3GPP) Home Page. Available at http://www.3gpp.org

[84] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, Special issue on Mobile Ad Hoc Networking: Research, Trends, and Applications, 2(5), 2002.

[85] D. T. Suzuki, A. J. F. Griffiths, J. H. Miller, and R. C. Lewontin, *Introduction to Genetic Analysis*, 4th ed., W. H. Freeman, 1989.

[86] The Internet Engineering Task Force Home Page. Available at http://www.ietf.org

[87] Session Initiation Protocol (SIP) Home Page. Available at http://www.cs.columbia.edu/sip

[88] RFC 3548, "The Base16, Base32, and Base64 Data Encodings." Available at http://www.ietf.org/rfc/rfc3548.txt

[89] RFC 3339, "Date and time on the Internet: Timestamps." Available at http://www.ietf.org/rfc/rfc3339.txt

[90] B. Kalinski, "RSA Digital Signature Standards," 23rd National Information Systems Security Conference, 2000.

[91] RSA Security Home Page. Available at http://www.rsasecurity.com

[92] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2003.

[93] DSA, RSA, ECDSA (FIPS 186-2) Standards. Available at http://crsc.nist.gov/cryptval/dss.htm

[94] Standards for Efficient Cryptography group, "SEC1: Elliptic curve cryptography," 2000. Available at http://www.secg.org

[95] Standards for Efficient Cryptography group, "SEC2: Recommended elliptic curve domain parameters," 2000. Available at http://www.secg.org

[96] L. Guibas and R. Sedgewick, "A dichromatic framework for balanced trees," 19th Annual IEEE Symposium o Foundations of Computer Science (FOCS), 1978.

[97] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, 35(4), 1988.

[98] B. V. Cherkassky and A. V. Goldberg, "On implementing the push-relabel method for the maximum flow problem," *Algorithmica*, 19(4), 1997.

[99] Electronic Frontier Foundation (EFF) Wireless friendly ISP list. Available at http://www.eff.org/Infrastructure/Wireless_cellular_radio/wireless_friendly_isp_list.html

[100] British Telecommunications, 8 Mbps Digital Subscriber Line (DSL). Available at http://www.faster.bt.com

[101] The FON Home Page. Available at http://www.fon.com