UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Managing Incentives in Community Network Clouds

Muhammad Amin Khan

Supervisors: Doctor Felix Freitag
Doctor Luís Eduardo Teixeira Rodrigues

Thesis approved in public session to obtain the PhD Degree in
Information Systems and Computer Engineering
Jury final classification: Pass with Distinction

Jury

Members of the Committee:
Doctor Luís Eduardo Teixeira Rodrigues
Doctor Llorenç Cerdà-Alabern
Doctor Joan Manuel Marquès i Puig
Doctor Gianfranco Giulioni
Doctor Ricardo Jorge Feliciano Lopes Pereira

2016

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Managing Incentives in Community Network Clouds

Muhammad Amin Khan

Supervisors: Doctor Felix Freitag
Doctor Luís Eduardo Teixeira Rodrigues

Thesis approved in public session to obtain the PhD Degree in
Information Systems and Computer Engineering
Jury final classification: Pass with Distinction

Jury

Members of the Committee:

Doctor Luís Eduardo Teixeira Rodrigues, Full Professor, Instituto Superior Técnico, Universidade de Lisboa

Doctor Llorenç Cerdà-Alabern, Associate Professor, Universitat Politècnica de Catalunya, Spain

Doctor Joan Manuel Marquès i Puig, Associate Professor, Universitat Oberta de Catalunya, Spain

Doctor Gianfranco Giulioni, Associate Professor, Università degli Studi "G. d'Annunzio" Chieti-Pescara, Italy

Doctor Ricardo Jorge Feliciano Lopes Pereira, Assistant Professor, Instituto Superior Técnico, Universidade de Lisboa

Funding Institutions

European Commission (EACEA) through Erasmus Mundus Joint Doctoral fellowship

2016

To my family.

To my teachers, my mentors.

Acknowledgments

I am hugely indebted to my advisors, Felix Freitag and Luís Rodrigues, without whose guidance and support, this journey would never have been possible.

I would also like to thank my co-authors Ümit C. Büyükşahin (UPC), Roger P. Centelles (Guifi), Emmanouil Dimogerontakis (UPC), Jacek Dominiak (CA Labs), Smrati Gupta (CA Labs), Mennan Selimi (UPC), Leila Sharifi (IST), and Xavier Vilaça (INESC-ID), who were a great pleasure to work with. I am also thankful to my colleagues at Distributed Systems Groups, DSG at UPC and GSD at INESC-ID, who provided a great work environment, and the backdrop from many an insightful discussions.

During the course of my research, I got a chance to interact with many great minds, and I appreciate their feedback and insights, notably Jörn Altmann (Seoul), Roger Baig (Guifi), Gianfranco Giulioni (Chieti-Pescara), Leonardo Maccari (Trento), Victor Muntes (CA Labs), Leandro Navarro (UPC), Navaneeth Rameshan (UPC), Omer F. Rana (Cardiff), Davide Veiga (UPC), and Luis Veiga (INESC-ID, IST). I would like to especially thank Ricardo Pereira (INESC-ID, IST) who reviewed an earlier draft of this manuscript and provided valuable suggestions.

On this life long journey of learning, I am lucky to be guided and mentored by many great *teachers* on its many twists and turns, and I owe to them my contributions and achievements. Lastly, I am thankful to my family for their tremendous support.

* * *

This work was funded by European Commission (EACEA) through the Erasmus Mundus doctoral fellowship, via Erasmus Mundus Joint Doctorate in Distributed Computing (EMJD-DC) programme. This work was also supported by European Community Framework Programme 7 FIRE Initiative projects Community Networks Testbed for the Future Internet (CONFINE), FP7-288535, and CLOMMUNITY, FP7-317879. Support was also provided by the Universitat Politècnica de Catalunya BarcelonaTECH and the Spanish Government under contract TIN2013-47245-C2-1-R.

Abstract

Internet and communication technologies have lowered the costs for communities to collaborate, leading to new services like user-generated content and social computing, and through collaboration, collectively built infrastructures like community networks have also emerged. While community networks focus solely on sharing of network bandwidth, community network clouds extend this sharing to provide for applications of local interest deployed within community networks through collaborative efforts to provision cloud infrastructures. Community network clouds complement the traditional large-scale public cloud providers similar to the model of decentralised edge clouds by bringing both content and computation closer to the users at the edges of the network. Community network clouds are based on the principle of reciprocal sharing and most of their users are moved by altruistic principles. However, as any other human organisation, these networks are not immune to overuse, free-riding, or under-provisioning, specially in scenarios where users may have motivations to compete for scarce resources. We focus in this thesis on the incentives based resource regulations mechanisms to derive practical ways of implementing arbitration when such contention for limited resources occurs. We first design these regulation mechanisms for the local level where stronger social relationships between the community members imply trust, and ensure adherence to the system policies. We next extend the mechanisms for larger communities of untrusted users, where rational users may be motivated to deviate for their personal gains, and develop a distributed framework for guaranteeing trust in the resource regulation. Such mechanisms assist in encouraging contribution by the community members, and will help towards adoption, sustainability, and growth of the community cloud model.

Keywords

community cloud; community networks; cloud computing; economic mechanisms

Resumen

El Internet y las tecnologías de la comunicación han bajado los costos de colaborar en comunidad, dando lugar a nuevos servicios, como los contenidos generados por usuarios y la informática social y, por medio de la colaboración, han surgido infraestructuras construídas colectivamente, como las redes comunitarias. Mientras las redes comunitarias se centran exclusivamente en el intercambio de ancho de banda de la red, las nubes comunitarias extienden este intercambio para proporcionar aplicaciones de interés local, desplegadas en las redes comunitarias a través de actividades de colaboración para proveer infraestructuras en la nube. Las nubes comunitarias complementan a los proveedores tradicionales de la nube a gran escala, en un modo similar al modelo de las nubes descentralizadas, trayendo tanto el contenido como la computación más cerca hacia los usuarios en los extremos de la red. Las nubes comunitarias se basan en el principio de compartir recíprocamente y la mayoría de sus usuarios son movidos por principios altruistas. Sin embargo, como cualquier otra organización humana, estas redes no son inmunes al uso excesivo, al parasitismo, o al bajo-aprovisionamiento, especialmente en escenarios donde los usuarios pueden estar motivados a competir por recursos escasos. Nos centramos en esta tesis en los mecanismos de regulación de recursos basados en incentivos para derivar formas de aplicación práctica del arbitraje cuando se produce tal contención por recursos limitados. Primero diseñamos estos mecanismos de regulación a nivel local, donde las fuertes relaciones sociales entre los miembros de la comunidad generan confianza y aseguran la adhesión a las políticas del sistema. A continuación, extendemos los mecanismos para comunidades más grandes de usuarios no confiables, donde usuarios racionales pueden ser motivados a desviarse por sus ganancias personales, y desarrollamos un marco distribuido para garantizar confianza en la regulación de recursos. Tales mecanismos ayudan a fomentar la contribución de los miembros de la comunidad, y ayudan a la adopción, la sostenibilidad y el crecimiento del modelo de nube comunitaria.

Palabras Clave

nube comunitaria; redes comunitarias; computación en la nube; mecanismos económicos

Resumo

A Internet e as tecnologias de comunicação têm reduzido os custos para comunidades colaborarem, levando a novos serviços como conteúdo gerado pelos utilizadores e computação social, surgindo também, através de colaboração, infraestruturas construídas colectivamente, tais como redes comunitárias. Enquanto as redes comunitárias focam-se unicamente na partilha de largura de banda, as núvens comunitárias alargam esta partilha para providenciar aplicações de interesse local, implementadas dentro de redes comunitárias através de esforços colaborativos para providenciar infraestruturas em núvem. As núvens comunitárias complementam os tradicionais fornecedores de núvens públicas de larga escala, de forma similar ao modelo de núvens de limite, trazendo tanto conteúdo como computação para mais perto dos utilizadores nos limites da rede. As núvens comunitárias são baseadas no princípio de partilha recíproca e a maioria dos seus utilizadores são movidos por princípios altruístas. Contudo, tal como qualquer outra organização humana, estas redes não são imunes à sobreutilização, parasitismo, ou sub-provisão, especialmente em situações onde os utilizadores possam ter motivações para competir por recursos escassos. Nesta tese focamo-nos nos mecanismos de base para incentivo de regulação de recursos, para derivar formas práticas de implementar arbitragem quando ocorre disputa por recursos limitados. Primeiro projectamos estes mecanismos de regulação ao nível local, onde os laços sociais entre membros da comunidade são mais fortes e implicam confiança, e garantem adesão às políticas do sistema. Em seguida alargamos os mecanismos para comunidades de utilizadores não-confiáveis maiores, onde utilizadores racionais podem estar motivados a desviar-se do comportamento esperado para ganho pessoal, e desenvolvemos uma estrutura distribuída para garantir confiança na regulação de recursos. Tais mecanismos incentivam à contribuição dos membros da comunidade, e ajudando no sentido da adopção, sustentabilidade e crescimento do modelo de núvens comunitárias.

Palavras Chave

nuvem comunitária; redes comunitárias; computação em nuvem; mecanismos económicos

List of Publications

The research results from this thesis have led to the following publications:

Journal Articles

[KBF15] Amin M Khan, Umit Cavus Buyuksahin, and Felix Freitag. "Incentive-based resource assignment and regulation for collaborative cloud services in community networks". In: 81.8 (Dec. 2015). (JCR IF: 1.138, Q2), pp. 1479–1495.

Conference Proceedings

[Kha+16] Amin M Khan, Xavier Vilaça, Luis Rodrigues, and Felix Freitag. "A Distributed Auctioneer for Resource Allocation in Decentralized Systems". In: *36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016)*. (CORE Rank A). June 2016.

[Kha+15] Amin M Khan, Xavier Vilaça, Luis Rodrigues, and Felix Freitag. "Towards Incentive-Compatible Pricing for Bandwidth Reservation in Community Network Clouds". In: *12th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2015)*. Springer International Publishing, Sept. 2015.

[KBF14] Amin M Khan, Umit Cavus Buyuksahin, and Felix Freitag. "Prototyping Incentive-Based Resource Assignment for Clouds in Community Networks". In: *28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014)*. (Best Paper Award, CORE Rank B). IEEE, May 2014, pp. 719–726.

[KSF14] Amin M Khan, Mennan Selimi, and Felix Freitag. "Towards Distributed Architecture for Collaborative Cloud Services in Community Networks". In: *6th International Conference on Intelligent Networking and Collaborative Systems (INCoS 2014)*. IEEE, Sept. 2014.

[KBF13] Amin M Khan, Umit Cavus Buyuksahin, and Felix Freitag. "Towards Incentive-based Resource Assignment and Regulation in Clouds for Community Networks". In: *Economics of Grids, Clouds, Systems, and Services*. Ed. by Jörn Altmann Altmann, Kurt Vanmechelen, and Omer F. Rana. Vol. 8193. Lecture Notes in Computer Science. Springer International Publishing, Sept. 2013, pp. 197–211.

Other Publications

The background research to this thesis has led to the following publications:

Book Chapters

[KFN16] Amin M Khan, Felix Freitag, and Leandro Navarro. "Community Clouds". In: *Encyclopedia of Cloud Computing*. Ed. by San Murugesan and Irena Bojanova. Wiley-IEEE, June 2016.

[Kha+16] Amin M Khan, Felix Freitag, Leandro Navarro, and Roger Baig. "Enabling Clouds in Community Networks". In: *European Project Space on Research and Applications of Information and Communication Systems*. Ed. by Carlos Cerqueira and James Uhomoibhi. SCITEPRESS, 2016.

Journal Articles

[Sel+15] Mennan Selimi, Amin M Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. "Cloud services in the Guifi.net community network". In: 93.P2 (Dec. 2015). (JCR IF: 1.256, Q2), pp. 373–388.

[KF14] Amin M Khan and Felix Freitag. "Sparks in the Fog: Social and Economic Mechanisms as Enablers for Community Network Clouds". In: 3.8 (2014).

Conference Proceedings

[Kha+15] Amin M Khan, Felix Freitag, Smrati Gupta, Victor Muntés-Mulero, Jacek Dominiak, and Peter Matthews. "On Supporting Service Selection for Collaborative Multi-Cloud Ecosystems in Community Networks". In: *29th IEEE International Conference on Advanced Information Networking and Applications (AINA 2015)*. (CORE Rank B). Mar. 2015.

[KFR15] Amin M Khan, Felix Freitag, and Luis Rodrigues. "Current Trends and Future Directions in Community Edge Clouds". In: *4th International Conference on Cloud Networking (CloudNet 2015)*. IEEE, Oct. 2015.

[KF14] Amin M Khan and Felix Freitag. "Exploring the Role of Macroeconomic Mechanisms in Voluntary Resource Provisioning in Community Network Clouds". In: *11th International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2014)*. Vol. 290. Advances in Intelligent Systems and Computing. Springer International Publishing, June 2014, pp. 269–278.

Workshops

[Fre+14] Felix Freitag, Leila Sharifi, Amin M Khan, Leandro Navarro, Roger Baig, Pau Escrich, and Luis Veiga. "A Look at Energy Efficient System Opportunities with Community Network Clouds". In: *Workshop on Energy-Efficient System (EES), within 2nd International Conference on ICT for Sustainability (IST4S 2014)*. Aug. 2014.

[BKF13] Umit Cavus Buyuksahin, Amin M Khan, and Felix Freitag. "Support Service for Reciprocal Computational Resource Sharing in Wireless Community Networks". In: *5th International Workshop on Hot Topics in Mesh Networking (IEEE HotMESH 2013), within IEEE WoWMoM*. (CORE Rank C). Madrid, Spain: IEEE, June 2013.

[Kha+13] Amin M Khan, Leila Sharifi, Leandro Navarro, and Luis Veiga. "Clouds of Small Things: Provisioning Infrastructure-as-a-Service from within Community Networks". In: *2nd International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2013), within IEEE WiMob*. IEEE, Oct. 2013, pp. 16–21.

Abstracts, Demos & Posters

[Bai+15] Roger Baig, Felix Freitag, Amin M Khan, Agusti Moll, Leandro Navarro, Roger Pueyo Centelles, and Vladimir Vlassov. "Community Clouds at the Edge deployed in Guifi.net". In: *4th International Conference on Cloud Networking (CloudNet 2015)*. IEEE, Oct. 2015.

[Sel+14] Mennan Selimi, Jorge L Florit, Davide Vega, Roc Meseguer, Ester Lopez, Amin M Khan, Axel Neumann, Felix Freitag, Leandro Navarro, Roger Baig, Pau Escrich, Agusti Moll, Roger Pueyo Centelles, Ivan Vilata, Marc Aymerich, and Santiago Lamora. "Cloud-Based Extension for Community-Lab". In: *22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2014)*. (CORE Rank A). Paris, France: IEEE, Sept. 2014, pp. 502–505.

[Jim+13] Javi Jiménez, Roger Baig, Pau Escrich, Amin M Khan, Felix Freitag, Leandro Navarro, Ermanno Pietrosemoli, Marco Zennaro, Amir H Payberah, and Vladimir Vlassov. "Supporting cloud deployment in the Guifi.net community network". In: *5th Global Information Infrastructure and Networking Symposium (GIIS 2013)*. (CORE Rank C). IEEE, Oct. 2013.

Technical Reports

[KBF13] Amin M Khan, Umit Cavus Buyuksahin, and Felix Freitag. *Distributed Architecture for Cloud System tailored for Wireless Community Networks*. Tech. rep. UPC-DAC-RR-XCSD-2013-4. Barcelona, Spain: Universitat Politècnica de Catalunya, May 2013.

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

Recent developments in communication technologies like the Internet, email and social networking have significantly removed the barriers for communication and coordination for small to large groups, bringing down the costs that obstructed collaborative production before the era of the Internet. The ICT revolution ushered in group communication and collaborative production with popular applications now widely adopted, like social networking, social bookmarking, user-generated content, photo sharing, and many more. Even infrastructures based on a cooperative model have been built, for example community wireless mesh networks [Bra+13] gained momentum in the early 2000s in response to limited options for network connectivity in rural and urban communities.

Community networks represent a social collective to build ICT infrastructures for serving interests of the rural and urban communities. Volunteers in their local communities use off-the-shelf network equipment and open unlicensed wireless spectrum to provide network and communication services. These community networks have been popular, and have recently also employed fibre optic links [Bai+15b], for example Athens Wireless Metropolitan Network (AWMN) [Ath16], Freifunk [Frei16], FunkFeuer [Fun16], Guifi.net [Gui16], and Ninux [Nin16], are some of the networks deployed in Europe, having up to 28,000

nodes [Bai+15b]. Community networking thus presents an emerging model for the Future Internet across Europe and beyond, allowing for communities of citizens to build, operate and own open IP-based networks, and enabling individual and collective digital participation.

Community networks are based on the concept of reciprocal sharing, but this sharing is limited to network bandwidth, and does not extend to other computing resources. Community clouds aim to address this limitation, enabling the sharing of all types of computing resources, following the model of cloud computing, and assist in developing services and applications of local interest within community networks. Community cloud in this context refers to the cloud hosted on community-owned computing and communication resources providing services of local interest.

Community clouds, similar to community networks, are based on volunteer efforts, so need to provide tangible or intangible benefits to the users in order to keep them engaged. This requires that resource allocation mechanisms are designed to incentivise contribution from the users, so as to enable the community cloud transition from inception through early adoption to finally ubiquitous usage, leading to a sustainable and viable ecosystem.

1.1 Problem Statement

The problem of allocating shared resources efficiently and effectively is a challenging one, with the need to provide various guarantees like to have maximal social welfare (better utility for all the participants), to ensure truthfulness from the users (so they do not have incentives to lie about their requests), and, at the same time, to be computationally efficient. This is particularly important in the case of community network clouds where there is no centralised entity responsible and in control of all the resources available in the community network.

The main objective of this thesis is to develop resource regulation mechanisms to incentivise contribution and maximise the utility of the community cloud system, while guaranteeing trust and ensuring truthfulness from its participants.

1.2 Research Methodology

The aim of this thesis is to devise practical and effective incentive mechanisms for community network clouds, and we approach this mainly by the following means:

- **Theoretical proofs**. We theoretically prove the soundness of our proposed framework.
- **Simulation evaluation**. We conduct simulation experiments, to better understand the behaviour of our mechanisms at scale, and to demonstrate their applicability.
- **Prototype deployment**. We implement and deploy prototype components in a testbed, to get insights within the real-world constraints of the community networks.

1.3 Results

In this thesis, we present the complete framework of a community cloud system, in particular focusing on the resource allocation components. Figure 1.1 shows how we approach this framework, and we include the chapters and sections where we discuss different aspects of the framework. We discuss the overall architecture in Chapter 3. We present the vision behind the community cloud system in § 6.1.1, and discuss the relevance of the social and economic context of community networks in devising mechanisms to drives the adoption of the community network cloud in § 6.1.2.

At the core layer, we study how scalability affects the design of community cloud system in § 6.1.3. Within middleware layer, we emphasise the resource regulation components. For a community of local users with strong social ties ensuring trust, we propose resource regulation mechanisms in Chapter 4. For larger communities which lack trust among the users, we propose distributed auctioneer in Chapter 5. Another service within middleware layer is a decision support system (DSS) to facilitate selection of services, which we touch upon in § 6.1.4. Within services layer, we need applications that provide utility for the members of the community network, which we discuss in § 6.1.5.

1.4 Summary of Contributions

The contributions of this thesis include:

1. For local communities of trusted users, this thesis proposes incentive-based resource regulation mechanisms. These mechanisms are computationally efficient and incur minimal overhead, and ensure contribution from the participants for the viability of the community cloud system.
2. For a community of untrusted users, this thesis proposes distributed virtual auctioneer using a novel distributed framework for devising Nash equilibria distributed simulations of the auctioneer

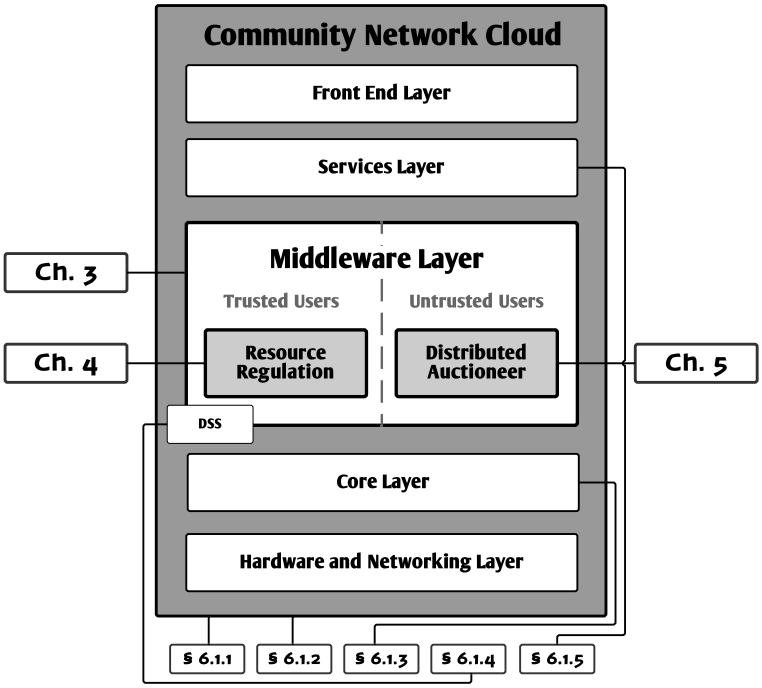


Figure 1.1: Overview of the proposed framework

that are resilient to asynchrony and coalitions. This can be used as a building block to implement resource allocation in a decentralised community cloud system.

1.5 Outline of the Thesis

This thesis is organised as follows. Chapter 2 presents the state-of-the-art. Chapter 3 discusses the middleware for resource regulation in community network clouds. Chapter 4 proposes incentive mechanisms for a community of trusted users. Chapter 5 devises a framework for managing incentives with untrusted users. Chapter 6 concludes the thesis and indicates future directions.

2

State-of-the-Art

Community network cloud is a social collective model, and needs contribution from its participants for its sustainability and growth. This requires addressing the challenging and concrete problem of effective resource regulation. In this chapter, we present an overview of the state-of-the-art in resource regulation, and study the role of the incentives and trust in ensuring users' participation.

2.1 Definitions

We start by introducing the terms used throughout this chapter and the rest of the thesis.

Community Network Cloud Community network clouds get formed when cloud services are provisioned using resources contributed by the members of the community network to build services and applications of local interest. In this thesis, we often use the term community clouds interchangeably for community network clouds, unless otherwise stated which will be clear from the context.

Dominant Strategy A strategy is dominant for a player when it provides better utility to her than any other strategy, no matter what strategy other players choose.

Efficiency Efficiency refers to an increased aggregate valuation for all the users. An optimal solution maximises the social welfare in the system.

Ex Post Budget Balance Budget balance is the property that after the resources have been allocated the total value paid by users covers the total payments made to the providers.

Ex Post Nash Equilibrium Ex post Nash equilibrium means that the system reaches a Nash equilibrium no matter what a scheduling adversary does.

Fairness A fair resource allocation assigns each user a share of system resources which is close to its share of total system funding. In other words, fairness refers to minimising envy between the users resulting from allocation of resources.

Free Riding Free riding refers to the situation where users consume public goods or resources without any or minimal contributions from their side, which results in under-provision of these goods or resources.

Incentive Compatibility Incentive compatibility means that there is no incentive for any player to lie about her private information.

Individual Rationality Individual-rationality means that every player truthfully participating in the allocation is expected to gain no less utility than by not participating.

Nash Equilibrium A joint strategy involving all players in a game is a Nash equilibrium if no player can achieve better utility by unilaterally deviating, provided all the other players continue to play their role as suggested by the joint strategy.

Social Welfare Social welfare is an aggregate over the utility of all the players in an allocation.

Solution Preference Solution preference means that no player has the incentive to fail the algorithm. Even though players have different preference over the outcomes of an algorithm, no player can gain if the algorithm fails and there is no solution.

Strategy Proofness Strategy proofness means that the dominant strategy for all the players is to report their true valuations to the allocator, and therefore no player has any incentive to cheat.

Utility Utility function in game theory measures preference over the allocation of resources, and represents satisfaction experienced by the players.

Vickrey-Clarke-Groves (VCG) Mechanism VCG is a generic truthful mechanism for achieving a socially-optimal solution [NR99]. It is a generalisation of a Vickrey–Clarke–Groves auction, where each individual is charged a "social cost" equivalent to the harm they cause to other bidders.

Virtual Machines Virtual machine (VM) is an emulation of a computer system, where hardware resources, like computing, memory, and storage, are packaged as virtual instances.

2.2 Incentives

Many volunteer and distributed platforms appeal to users' altruistic instincts. Projects like SETI@Home [And+02], BOINC [And04], and Folding@home [Beb+09] propose to solve challenging scientific problems, which encourage users to contribute the idle resources of their machines. Other projects like PlanetLab [Chu+03] and various Grid systems [FK03] require some mutually-agreed upon level of contribution as a pre-condition for participating in the system.

Other earlier popular peer-to-peer (P2P) file sharing programs like Kazaa, Gnnutella, Napsters, and eMule among others [RD10], allowed users to upload and download files for free, and often suffered from the issues like free riding, under-provisioning, etc. This has led to designing incentive mechanisms in P2P systems to ensure that users actively contribute. For example, BitTorrent, using reciprocity principle, only allows users to download content if they also upload part of it to other BitTorrent users. Similar incentive mechanisms have been extensively studied for other P2P systems [BCF07; She+10; ZAM10].

Community networks are also based on the principle of reciprocal sharing, and offer various tangible and intangible benefits to their users. Bina and Giaglis [BG06] have explored various psychological and

social motivations of the users of AWMN community network. Reciprocal resource sharing is, in fact, part of the membership rules or peering agreements [Pico5] of many community networks. The Wireless Commons License (WCL) [Wir10] of many community networks states that the network participants that extend the network, e.g. contribute new nodes, will extend the network in the same WCL terms and conditions, allowing traffic of other members to transit on their own network segments. Therefore, resource sharing in community networks from the equipment perspective refers in practice to the sharing of the nodes' bandwidth. This sharing, done in a reciprocal manner, enables the traffic from other nodes to be routed over the nodes of different node owners, and allows community networks to successfully operate as IP networks.

Most of these incentive mechanisms are based on the idea of reciprocating individual's *contribution*, anybody who contributes more value to the system is allowed to reap more benefits from it. However, this does not take into account that in many cases not all users are as rich in resources as others. Participatory Economics (Parecon) model envisions rewarding the users based on their effort, which is defined as contribution as a fraction of their capacity, instead of rewarding purely on the basis of their absolute contribution [Albo4]. Effort-based incentives [Rah+10; Veg+13; Veg15] have been proposed based on Parecon principle to achieve fairness and improve social welfare, while better addressing the resource heterogeneity in the system. For instance, Rahman et al. [Rah+10] apply effort-based incentives for file sharing in BitTorrent where some users with slow Internet connections cannot upload as much content as others. When deciding how much a user can download, they propose factoring in user's connection speed in addition to the data the user has uploaded. Their results show that the effort-based mechanism remains incentive-compatible and improves efficiency and fairness as compared to purely contribution-based approach. We also turn to effort-based incentives in Chapter 5 when devising mechanisms for resource sharing between the users of community network clouds.

2.3 Economic Based Resource Allocation

Most distributed systems have limited resources that need to be shared by many nodes. For instance, in a network there is limited bandwidth that needs to be allocated to multiple nodes. In cloud applications, virtual machines (VMs) need to be allocated to different cloud users. Resource allocation is, therefore, a key problem in distributed systems, and there is a vast literature on resource allocation of shared resources, whether they be network bandwidth, or other general computing resources.

Distributed resource allocation is particularly challenging when nodes operate under different spheres of control and may not be willing to cooperate. Namely, a resource allocation strategy that assumes that all nodes execute a given algorithm may break if nodes may extract benefits by deviating from the expected behaviour. Many examples of this problem can be found in the literature. The works of [Lee+07] and [Xia+13] illustrate how a network user may attempt to monopolise the bandwidth utilisation if it has the opportunity. There is evidence that programmers can instrument their code to get an unfair advantage of several Unix schedulers [GC05]. In shared infrastructures, like Grid systems, participating users try to maximise their own usage to the detriment of the others [Lai+04]. Dynamic wireless spectrum allocation suffers from unfair manipulation [Zho+08]. Social cloud computing [Cat+14] and cooperative computing systems like BitTorrent [Liu+10] suffer when users act selfishly in consuming resources.

An approach that has emerged as a viable alternative for the problem above is to use economic models to address resource allocation, in particular by resorting to auction systems [Hur73; RS81]. As a result, an extensive literature exists on the use of several types of auctions to perform resource allocation in distributed systems [NR99; Niy13; Wal+92]. In particular, the advent of the cloud computing model, where many clients may compete for the resources managed by one or more providers, has spurred the usage of different auction mechanisms for the cloud [WRM12].

In these approaches, users are modelled as non-cooperative rational players who are willing to pay for using resources or get paid for providing those resources. Specifically, users declare to an auctioneer the preference for different allocations of resources, and the auctioneer executes some auction mechanism to derive an allocation between users and resources that maximises social welfare (preferences of users for the allocation), and the payments to be performed or received by each user. The aim is to obtain an allocation with a social welfare as close as possible to the optimal while ensuring truthfulness from the users, such that they do not have incentives to lie about their bids. In addition to maximal social welfare and truthfulness, other guarantees may be provided, including computational efficiency and budget balance (the payments made by the users outweigh the payments received).

In the following, we look at the state-of-the-art in economic-based resource regulation, first for community networks (§ 2.3.1), and then for cloud systems (§ 2.3.2) both in the context of virtual machines and network bandwidth. Next, we study managing resource sharing between federations of cloud providers (§ 2.3.3) .

2.3.1 Community Networks

Various pricing schemes, game theoretic mechanisms and auction-based approaches for arbitrating network resources have been proposed and used in practice. Maillé and Tuffin [MT14] provide a comprehensive survey of the historical approaches and the current state-of-the-art from the viewpoint of telecommunication services.

Community networks comprise of wireless mesh and multi-hop networks, and require cooperation among its users for proper functioning. Game theory, both non-cooperative and cooperative, has been applied in literature for studying the incentives of its user and devising network allocation mechanisms. Xiao et al. [Xia+13] propose a general framework for studying the user cooperation in a network, and formalise the relationship between incentive, fairness and efficiency for cooperative networks. Zhou et al. [ZLL14b] develop a Vickrey-Clarke-Groves (VCG) based mechanism for non-cooperative users. Since VCG mechanism cannot be directly applied because of the high computational complexity, they modify it by using relaxation-based greedy algorithm in such a way that it still guarantees strategy-proofness and efficiency. Lee et al. [Lee+07] focus on the problem of backbone construction with selfish users in a community network, who want others to relay their packets but want to avoid relaying packets for others. They propose an incentive-compatible protocol based on Volunteer's Timing Dilemma from non-cooperative game theory. Their findings show that using their protocol the backbone forms quickly, with characteristics comparable to protocols designed only for altruistic users.

Community networks like Guifi.net, in practice, also put emphasis on social sharing agreements [Pico5; Wir10], and when conflicts occur enforce these agreements through social mechanisms [Bai+15b]. For our context of community network clouds, we accept the guarantees implied by the community networks, and assume that the network is owned by the whole community so the traffic within the network between any two nodes is ensured to transit over intermediary nodes. Therefore, when we address the issue of bandwidth reservation in Chapter 5, we focus solely on the gateways present in the community network that provide access to the Internet.

2.3.2 Cloud Systems

With respect to the general computing resources, there has been a lot of existing work in the context of Grid systems [FKo3; BAV05; CW12] and shared infrastructures like PlanetLab [Leo13]. For instance, one of the system implemented for PlanetLab is Tycoon [Lai+04], which is a distributed marked-based

resource allocation system, implementing proportional fairness using decentralised isolated auctions. Tycoon allows users to differentiate their jobs based on their importance by specifying different bid amounts. Auctioneers manage only local resources, and users submit separate bids to these auctioneers. The bids remain valid until a user's credit gets low, so the mechanism reduces manual bidding overhead by the users. Resources are assigned in proportion to the bid amounts using a best-effort approach. This allows Tycoon to achieve efficient usage of resources, while maintaining little allocation overhead.

Economic-based resource allocation mechanisms have been extensively explored for cloud computing [NFL12; Pop+12; Shi+14; WRM12; Zha+13; Zha+15b; Zhe+14; Zhe+15]. Amazon, one of the leading providers of public cloud services, was among the first to offer cloud resources using market-driven prices [Agm+13]. Besides investigating how a provider can use economic-based mechanisms to maximise the utilisation of its resources and increase its revenue [WRM12], the recent research has also looked into devising the best strategies for the users to bid for the cloud resources in such a market [Zhe+15]. Many works, for instance [ZLW14; Zha+15b], employ the celebrated Vickrey-Clarke-Groves (VCG) mechanism [NR99] for achieving truthfulness and a trade-off between maximal social welfare and computational efficiency. The challenge in applying VCG mechanisms in clouds is that in most of these problems, finding an optimal allocation is NP-hard, and so traditional VCG mechanism cannot be applied. One key line of work has been to use randomised algorithms [ZLW14; Zha+15b], or linear programming decomposition techniques [NL13; Zha+15a], to achieve strategy-proofness with a computationally feasible solution, albeit achieving less than optimal social welfare. The overall aim is to look for solutions with lower computational complexity, and higher social welfare, while ensuring strategy-proofness.

Virtual Machines Allocation

Cloud computing employs virtualization to package computing resources like CPU time, memory, and storage as virtual machines (VMs). Therefore, resource allocation in cloud, for the most part, deals with composing VMs from the hardware resources, and allocating them to the users in the most efficient manner. Cloud providers offer a variety of VM instances of different types, where type refers to the composition of different resources packaged in the VM instance. For example, a VM consisting of 2 virtual CPU units, 8 GB RAM and 100 GB storage is one type of VM. Zhang et al. [Zha+15b] extend this to allow users to request customised dynamically assembled VM types, bundling VMs from different geo-distributed data centres of the provider. They use smoothed analysis and randomised reduction to design

a randomised, highly efficient auction mechanism. Their mechanism is general and expressive enough to encompass various cloud scenarios, and achieves truthfulness (in expectation), polynomial running time (in expectation), and $(1 - \epsilon)$-optimal social welfare (in expectation) for resource allocation in a geo-distributed cloud, where $\epsilon \in (0, 1)$.

Zhang et al. [ZLW14] approach combinatorial auctions of heterogeneous VMs by modelling social welfare maximization as a mixed linear integer program. They design an efficient α-approximation algorithm, with $\alpha \sim 2.72$ in typical scenarios. They use this algorithm as a building block for designing a randomised combinatorial auction that is computationally efficient, truthful in expectation, and guarantees the same social welfare approximation factor α. They utilise a pair of tailored primal and dual linear programs (LPs) to decompose fractional solution of social welfare maximization problem into a convex combination of integral solutions.

Bandwidth Reservation

The focus remains mostly on efficiently allocating VMs in the cloud, even though the bandwidth, both upstream and downstream, to connect to VMs in the cloud is metered. Bandwidth allocation and reservation gains significance, in particular when the applications are network-intensive or have real-time constraints. Some prime examples are video-streaming, video-on-demand and cloud-based gaming. Recent work has explored various economic based bandwidth allocation schemes for public clouds [Gui+14; Guo+13; NFL12; Pop+12; SL14; Zhe+14]. The emphasis in the cloud has been on having bandwidth reserved with service level guarantees for the cloud applications.

Gui et al. [Gui+14] propose VCG-auction based mechanisms for reserving bandwidth at the multiple geo-distributed data centres of the cloud provider. The users submit bandwidth reservation requests separately for each of the data centres. In case the users can accept partially fulfilled requests, i.e. bandwidth reserved up to the maximum requested, a solution can be calculated using linear programming in polynomial time, which achieves both optimal social welfare and strategy-proofness. However, if partial reservations are not permissible, the allocation problem is NP-hard, so the above polynomial time algorithm cannot be used to calculate optimal allocation. The authors propose a heuristics-based greedy algorithm that guarantees strategy-proofness, though provides less than optimal social welfare. Zheng et al. [Zhe+14] focus on a multi-cloud scenario where users need to reserve bandwidth from different cloud providers, because they have strict requirements on the amount of bandwidth for guaranteeing their quality of ser-

vices. They model this open market of cloud providers as a double-sided auction, where providers also submit bids to the auctioneer besides the users, and propose strategy-proof mechanisms based on McAfee double auction [MS83].

Shen and Li [SL14] propose a bandwidth pricing model, a network bandwidth sharing policy and flow arrangement policies, and use non-cooperative game theory analysis. Their policies encourage cooperation among the tenants of the cloud infrastructure, who are incentivised to prefer uncongested links and constrain congestion. Guo et al. [Guo+13] focus on the bandwidth available within the data centre infrastructure, on the links connecting the multiple VMs owned by the tenants. They apply cooperative game-theoretic framework to design a distributed algorithm to achieve efficient and fair bandwidth allocation corresponding to the Nash bargaining solution.

2.3.3 Cloud Federations

Aside from public clouds, another emerging model in cloud computing involves cloud providers trading of VM resources among themselves, referred to as federating their individual clouds. There are various terms for this scenario in literature such as federated clouds, Intercloud, community clouds, cloud brokerage, etc. In this case, cloud providers agree to provision VMs for each other, for instance one provider can solicit additional resources from others to satisfy peaks in demands. Such a federated or community cloud presents the challenge of a free market, where participants have the incentive only to accept those resource exchanges that are profitable for them. Zhao at al. [ZLL14a] develop a distributed market-oriented model for the resource negotiation and trading problem in such a community cloud. They use this model to propose a multi-agent based approach that provides an efficient and fair resource allocation for a group of autonomous cloud providers. They also consider resource trading under budget constraints, and based on a directed hypergraph model, present effective heuristic-based distributed protocols to achieve resource allocation within budget limits.

Social cloud computing [Cat+14; Pun+13] similarly focuses on trading resources between cloud providers, who in this case are individual users of online social networks, like Facebook, Twitter, etc. Punceva et al. [Pun+13] propose a decentralised resource sharing model, and use virtual currency to incentivise cooperation without requiring a central reputation management system. They differentiate between intra-community and inter-community sharing, where a community consists of a group of "friends" on social networks, because the trust inherent within a tightly knit community aids in designing a more flexible

virtual currency representation. Caton et al. [Cat+14] focus more on improving how the providers are matched to the users in a social cloud. They propose heuristics based matching algorithms for bidirectional preference-based socially-aware resource allocation, with the aim to optimise social welfare and allocation fairness.

Li et al. [Li+13] study how individual cloud providers can maximise their profits through better resource trading and scheduling. They apply a double auction-based mechanism, which is strategy proof, individual rational, and ex-post budget balanced. Based on this auction mechanism, they propose an efficient and dynamic resource trading and scheduling algorithm, which carefully computes the true valuations of VMs in the auction, and aims to optimally schedule stochastic job requests onto the VMs. Palmieri et al. [Pal+13] focus on scheduling resources within federated clouds, and present a fully distributed agent-based game-theoretic scheme for scheduling computing resources between providers in federated clouds. Their scheme is based on independent, competing, and self-interested task execution agents, with the goal to achieve an optimum social welfare criteria towards a Nash equilibrium solution, using a slotted time model to provide advance reservation of resources in a fully distributed manner.

For the scenario of community network clouds, we consider members of community networks as cloud providers, and present efficient incentives based mechanisms for allocating cloud resources in Chapter 4.

2.4 Trust in Resource Allocation

The issue of trust in auctions is well-known and well-studied in economic theory, for various type of auctions [Sano0]. In particular Vickrey auction, which forms basis of the celebrated VCG mechanism [NR99], is very susceptible to a lying auctioneer [Sano0]. VCG mechanism is often used in distributed systems and cloud computing to ensure strategy-proofness in resource allocation, but it suffers from significant issues because of the trust required in the auctioneer [Sano0]. There are numerous distributed auction schemes proposed in literature [Guo+13; Zho+08; Lai+04], but the fact that they are decentralised in itself does not imply trust. Without careful design, one or more agents participating in the distributed auction can affect the results to their advantage.

The issue of an auctioneer cheating in second-price sealed-bid auctions, similar to VCG auctions, has been so severe that fraud was commonplace in the stamps auctions of late 19th and 20th century [Lucoob], which provide the first recorded example of using Vickrey auctions in practice. More recently, such second-price auctions have been used by eBay [Lucooa] to sell goods, and by Yahoo, Google, and other Internet

search companies to sell keywords-based online advertising [EOS07]. It has been suggested that such auctions are viable for these Internet companies, in the absence of any trust, because these companies have low commissions, and conduct so many trades that it is not in their interest to cheat [Lucoob]. But the issue of mistrust in the auctioneer has been prevalent, so much so that when Amazon introduced its "spot instances" service using market-based pricing, there were doubts that Amazon was not using supply and demand to set the prices, but instead employed a mathematical regression function to set the rates [Agm+13].

This lack of trust can create various problems for resource allocation in decentralised systems, and we go through a few examples here. InterCloud allows for multiple cloud providers to federate their resources to form a cloud market [GB14]. The current approaches mostly rely on bilateral agreements, and price negotiation. However, as the number of providers increases this arrangement will no longer be tenable. The providers cannot put absolute trust in anyone among them or in a third party to execute the auction fairly. Social clouds [Cat+14] allow exchanging resources between users of online social networks. Here, as well, the users have to trust the allocation and resource matching algorithm, which even if it runs distributed on multiple machines, cannot be trusted, since some of the users can *tweak* the allocator to their advantage. Secondary wireless spectrum markets [Zho+08; LLZ15] also apply auctions to ensure strategy-proofness, but again fail if a trusted auctioneer is not available. BitTorrent, like other similar popular P2P file sharing systems, regulates access to available resources depending on the users' contribution. However, users can easily use their BitTorrent clients to falsely report their contribution to be high. To counter this, BitTorrent private communities have a central mediator that dictates rules for uploading and downloading content, and tracks contribution and consumption by the users [Liu+10]. However, the administrators and privileged users can affect the central mediator to get unfair advantage. Resource allocation in shared infrastructures like Grid systems also require incentive-compatible solutions [Lai+04; KA06; Buy+02], and many of the proposed approaches in the literature apply only when a trusted entity is available for executing the auction mechanisms.

A cheating auctioneer can pose a number of problems. The major among them, that we focus on, is that the winning bids may not be selected fairly, and the payment each winning bidder has to make is not calculated properly. In open-bid auctions, like English and Dutch auctions [Kri09], the above two issues are not a major problem as the bidding process is open to all the participants. In sealed-bid first price auction, each winning bidder pays the amount she quoted, so even though the auctioneer can cheat on

selecting the winning bidders, at least the winning bidders are sure they are paying the correct amount. In sealed-bid second price auctions, the auctioneer can even cheat on the price the bidder has to pay, and this is a major issue [Sano0].

Other issues with auctions that we do not consider here include bidders cooperating together to artificially lower the price, the providers making a coalition to keep the price high, or the auctioneer learning from buyers' preference to increase the reserve bid price. We do not focus on these and other issues, but in recent literature cryptographic zero-knowledge based algorithms have been employed to tackle these problems [MR14; LAN03]. Some of these solutions, because of their high computational complexity, are better suited to infrequent auctions of highly valuable goods, for example wireless spectrum auction by a government, than for integrating in mechanisms for distributed systems where the auctions are often repeated at short intervals.

2.4.1 Allocation with Rational Users

Most of the existing approaches, as we have discussed above, assume a single trusted entity or multiple trusted entities coordinating together for executing the resource allocation mechanism. Unfortunately, this is an unreasonable assumption in many of today's fully decentralised systems, where all nodes are either resource consumers, resource providers, or both. In this case, there is no natural candidate that can be trusted by all other nodes to run the auction algorithm faithfully, given that any node may extract some benefit by perturbing the auction result. In some sense, all current distributed systems that rely on centralised auctions to perform resource allocation are not fully decentralised, because they depend on a unique central point of control, which is the trusted node that runs the auction algorithm. Similarly, in the context of community network clouds, that comprise of untrusted service providers, such a trusted entity in most cases is not feasible, and even if it existed, can impede the scalability. This leads to the observation that there is still a substantial gap that needs to be bridged to apply these results in fully decentralised settings. None of the above works, to the best of our knowledge, address the problem of resource allocation in the absence of a trusted auctioneer.

The problem of simulating the behaviour of a trusted entity in an environment with only rational players has been approached in the literature of distributed systems [Aiy+05; HT04; Abr+06; ADH13; Afe+14]. Aiyer et al. [Aiy+05] presented BAR model which involves Byzantine (or faulty) and rational players, as well as acquiescent players who follow the suggested protocols. They used BAR model to develop fault

tolerant cooperative services using state machine replication. In [HT04; Abr+06], the authors addressed the particular problems of secret sharing and multi-party computation assuming the existence of a trusted mediator, and then studied conditions under which it is possible to simulate the mediator through a distributed protocol. They discussed k-resilient Nash equilibria so that even if a whole coalition of up to k players chooses joint strategies to defect, still no member of the coalition can increase its utility. Their results apply even if there are only 2 players, so that multi-party computation can be performed with only two rational agents.

Abraham et al. [ADH13] devised k-resilient equilibria solutions for the classic problem of electing a leader in an anonymous network in the presence of rational agents. They considered existence of k-resilient equilibria for several topologies like a unidirectional ring, bidirectional ring, or completely connected network, in both the synchronous and asynchronous case. They showed that a fair ex post k-resilient equilibrium requires $n > 2k$, i.e. the number of agents should be more than twice the size of the largest coalition for the system to reach equilibrium. They also highlighted how involving cryptographic techniques can help achieve equilibrium, and result in more computationally efficient algorithms.

Afek et al. [Afe+14] proposed a building blocks approach for devising distributed k-resilient implementations to solve common distributed computing problems like leader election, consensus, and renaming. They extended the ideas from [ADH13] to develop building blocks that are all resilient in the presence of rational agents, and coalitions of rational agents to some extent as well. They differentiated between different utility functions for distributed algorithms, e.g., utilities based on communication preference, solution preference, and output preference. Based on these preferences, they formulated two basic building blocks for game theoretic distributed algorithms, a wake-up building block and and a knowledge sharing building block, that are resilient to these preferences. These blocks formed part of their solutions to leader election, consensus, and renaming problems.

None of these works devised distributed protocols for simulating the role of an auctioneer in an auction. We build on the work of [ADH13; Afe+14] to develop a distributed virtual auctioneer, which we focus on in Chapter 5.

3

Middleware for Resource Regulation in Community Clouds

Community clouds can have different interpretations depending upon the specific requirements and characteristics of the community, and how the infrastructures are deployed and services are provisioned. We focus, in our thesis, on the collaborative model of provisioning community cloud services based on volunteer computing paradigm as laid out in [MB09]. In this chapter, we explore the background of community clouds, and focus specifically on the problem of resource regulation. The chapter is organised as follows.

- We first look at the general idea of community clouds, and how they are used in the commercial sector (§ 3.1.1).
- Next we focus on citizen community clouds built collaboratively by the volunteers (§ 3.1.2).
- Then we study how the community networks provide an excellent context for deploying community clouds (§ 3.1.3).
- We present the overall architecture for realising a community cloud system (§ 3.2).

- We discuss different scenarios and contexts for applying resource regulation in a community network cloud(§ 3.3).

3.1 Community Network Clouds

The concept of community cloud computing has been introduced in its generic form [MG11] as a cloud deployment model in which a cloud infrastructure is built and provisioned for exclusive use by a specific community of consumers with shared concerns and interests, owned and managed by the community or by a third party or a combination of both. In this thesis, our focus is on the clouds built by the community, and for the community, relying on the resources available within community networks. We first discuss the general idea of community clouds, before going into the detail of community network clouds.

3.1.1 Commercial Community Clouds

Community cloud is one of the many different deployment models for cloud computing [MB16]. The most common and popular one, the public cloud, offers services of generic interest over the Internet, available to anybody who signs in with its credentials. On the other side, the private cloud model aims to provide cloud services to only a specific user group, such as a company, and the cloud infrastructure is isolated by firewalls avoiding public access. Finally, when a private cloud is combined with the public cloud, for instance some functionality of the cloud is provided by the public cloud and some remains in the private cloud, this is referred to as hybrid cloud. The community cloud [KFN16], bridges in different aspects the gap between the public cloud, the general purpose cloud available to all users, and the private cloud, available to only one cloud user with user-specific services.

Community clouds are implemented using different designs depending upon the requirements. One common approach is that a public cloud provider sets up separate infrastructure and develops services specifically for a community to provide a vertically integrated solution for that market. Similarly, a third party service provider can focus on a particular community and only specialise in building tailor-made solutions for that community. Another option is that community members that already have expertise in cloud infrastructure come together to federate their private clouds and collectively provision cloud services for the community.

Enterprises belonging to the same sector often use similar but independent cloud solutions, and comparing these solutions, it can be seen that these clouds are optimised in similar aspects which allow these

enterprises to gain advantages for reaching common goals. Instead of these private clouds, building a community cloud for such enterprises shares the cost of the cloud solutions among them, and may also offer collaborations for mutual benefit even among competitors. Such commercial community cloud solutions are a reality nowadays in several application areas and have been deployed in particular in the financial, governmental and health sector, besides many others, fulfilling the community-specific requirements, as evident from [Nys12; Opt12].

3.1.2 Citizen Community Clouds

Community clouds can also be built by the citizens in a bottom-up fashion through collaborative efforts [KFR15], by using resources contributed by individual users by either solely relying on user machines or using them to augment existing cloud infrastructures. The idea of building cloud infrastructure using resources contributed by a community of users [MB09], follows on from earlier volunteer distributed computing platforms like SETI@Home [And+02], BOINC [And04], Folding@home [Beb+09], HTCondor [TTL05], PlanetLab [Chu+03], and Seattle [Cap+09], among many others, and in general from the peer-to-peer systems [She+10] that focus on collaborative resource sharing.

There are a few such research prototypes for citizen community clouds that provide not the complete system but some of the components as a proof of concept. The Cloud@Home [DP12] project aims to harvest resources from the community for meeting the peaks in demand, working with public, private and hybrid clouds to form cloud federations. The P2PCS [BMT12] project has built a prototype implementation of a decentralised peer-to-peer cloud system, with basic support for creating and managing virtual machines using Java JRMI technology. The Clouds@home [Yi+11] project focuses on providing guaranteed performance and ensuring quality of service even when using volatile volunteered resources connected by Internet. Jang et al. [Jan+14] implement personal clouds that federate local, nearby and remote cloud resources to enhance the services available on mobile devices. Social cloud computing [Cha+12; Cat+14] takes advantage of the trust relationships between members of the online social networks to motivate sharing of storage and computation resources, by integrating with the programming interfaces (API) of the social network services which facilitates the establishment of mutual trust and resource sharing agreements.

3.1.3 Community Clouds in Community Networks

Community networks [Bra+13] are already based on the principle of sharing, though only of bandwidth, but the social aspects and community nature makes it easier to extend this sharing to other computing resources. The strong sense of community and technical knowledge of participants of such networks are some strong points which are conducive to building cloud applications tailored to local needs, built on infrastructure provided by the community members.

Community network clouds build on the success of community networks and aim to provide services and applications of local interest for the communities by applying the model of cloud computing. Community network clouds fit nicely with the recent shift in exploring alternative approaches to large-scale data centres based public cloud computing, which include Inter-Cloud and federated clouds (where multiple public cloud providers work together), hybrid clouds (where enterprises combine their own cloud infrastructure with the public clouds), and edge clouds using nano data centres [Sat+09] (where smaller clusters are deployed at the edges of the network to avoid latency and improve content-delivery). These initiatives provide an excellent backdrop to explore the role of the community network clouds in enhancing the value proposition of the community networks, since an infrastructure of nano data centres [Sat+09] to be deployed in a community network has to fit well with specific socio-economic and technical context of the community networks [KF14b]. For example, Guifi.net community network has deployed community edge cloud using their Debian-based Cloudy distribution [Bai+15a].

3.2 Architecture for Community Network Cloud

We foresee realising the community cloud by deploying a community cloud platform tailored to the specific infrastructure and context of community networks. A standard cloud system is usually a centralised platform designed to perform resource management. There are quite a few well known cloud platforms for managing public and private clouds, like OpenStack [Ope16b] and OpenNebula [Ope16a] among others. For community network cloud, we focus on providing a framework that would allow users to share resources and access collaboratively-built services in a distributed manner.

We propose a framework that can serve as the core of a community cloud system. Our community cloud framework is a distributed bottom-up resource sharing and collaborative services platform. This is achieved by adopting a layered architecture, as shown in Figure 3.1.

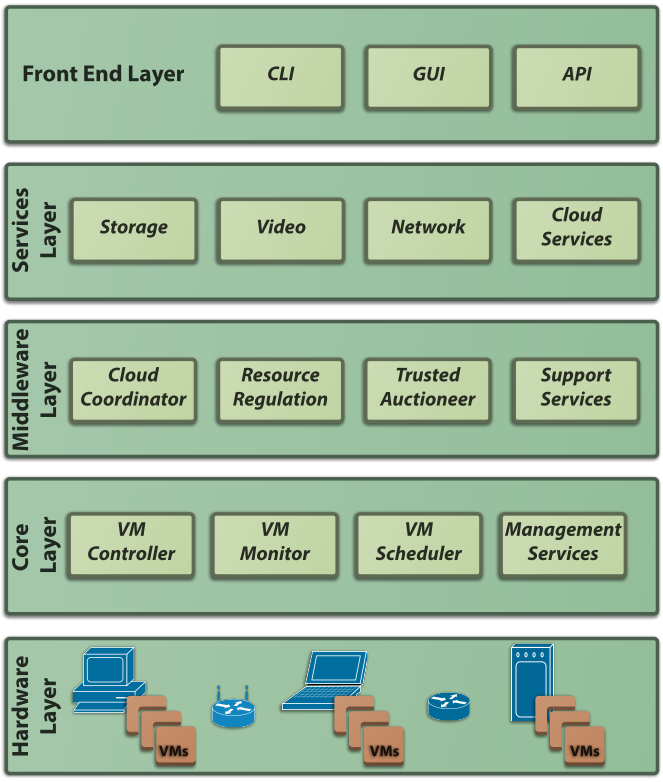


Figure 3.1: Framework for community cloud management system

1. The **Hardware layer** provides the physical infrastructure needed to run the cloud services and applications. The hardware in the community network customarily consists of management nodes, client nodes, routers and the communication infrastructure, along with any computation, storage and other resources attached to the nodes.

2. The **Core layer** is responsible for managing the hardware as virtualised resources. It consists of components, such as a manager for the hosts and the network as well as a controller, scheduler, monitor, and data storage for virtual machines (VMs).

3. The **Middleware layer** amalgamates the resources from multiple local community clouds, providing an integrated and consistent view of the cloud system to the cloud services. This can comprise of a variety of support services:

    - **Cloud coordinator** and services broker components for assisting in combining resources from multiple cloud providers.

- Incentives-based **resource regulation** and allocation components are important as a driver for users' participation for the sustainability and growth of the community cloud model. Our proposed incentive mechanisms in Chapter 4 can provide the building blocks for these components.
- **Trusted auctioneer** module for auction-based resource allocation schemes. Resource allocation component needs to be trusted by the users, and in turn the users need to follow the prescribed polices for the system to function properly. We return to this issue in Chapter 5, and devise a virtual distributed trusted auctioneer to integrate into the community cloud framework.
- Other **support services** may include, among many other:
  - Network coordination component to identify and manage different local clouds.
  - Service discovery component to keep track of the services provided by the various clouds.
  - Authentications and auditing components to support resource regulation.

4. The **Services layer** integrates useful services and applications providing utilities for the community network members to encourage their participation. Common services include storage, video streaming, video on demand, IP telephony, and network applications.
5. The **Front-end layer** provides the interface to interact with the infrastructure of the community cloud, including command line interfaces (CLI), graphical user interfaces (GUI), application programming interfaces (API), and any other tools for assisting in the development of cloud services and applications.

3.3 Incentives Based Resource Regulation

Community network clouds, like other volunteer computing platforms, not only need to solve the technical challenges to be feasible but also address the social and economic context to engage their participants. The existing social relationships among the users provide the foundation for building community cloud services. Just as in social computing [Cat+14], relationships from online social networks provide the backdrop to provision cloud services, in the case of community network clouds, we believe the existing relationships among users of the community networks will make it possible for the model to be adopted.

The community networks provide the context and motivating scenarios for developing the proposed solutions that are discussed in Chapters 4 and 5; however, our proposals and findings are general enough for being applicable to other related fields. In order to realise clouds in community networks, we can take the existing software and technology, and adapt and extend it to better fit with the characteristics and constraints of the community networks. For example, an initiative in this regard has been the Cloudy distribution [Clo16], which is based on Linux operating system and incorporates useful free and open-source services of interest to the members of Guifi.net [Sel+15]. Of course, this does not imply that the community cloud model is restricted only to the members of the community networks. In fact, with Cloudy distribution there has been an effort to engage and involve others who are not part of a community network, and let them connect using network tunnelling protocols [Com16a]. The emphasis is on building cloud services and applications of local interest using the resources contributed by the members of the community networks, but also on engaging the wider Internet community at the same time, which can also generate interest in the community networks themselves.

The resources available in the community network clouds can be divided into various categories. Network resources, like upstream and downstream bandwidth at different links of the community network, available bandwidth at the Internet gateways, or bandwidth at the links connecting to the popular content servers, are limited and already of value to the members of the community networks. Any community cloud solution has to work within the constraints of the available network capacity, and avoid negatively impacting the operations of the community network. Community clouds at the basic level provide Infrastructure-as-a-service (IaaS), where resources like computation, memory and storage, are packaged as virtual machine (VM) instances. In the collaborative model of community network clouds, users will be trading these resources as VMs among themselves. Other resources can include multimedia content, data storage, and other services, that are of interest to the users.

The underlying principles for regulating resources, in general, apply equally to all these different kinds of resources. In practice, the proposed solutions are often customised to better fit a particular problem scenario. In our work, we have focused on two scenarios, which provide broad coverage of resource regulation problem in the community network clouds, and our prototypes addressing the two scenarios bring completeness to the solution for managing incentives in community network clouds.

In Chapter 4, we focus on managing computing resources as VMs among the users. We propose efficient regulation mechanisms in the context of a small community of users, perhaps part of a single zone

of the community network, which already has trust and strong social relationships among its members. Our mechanisms regulate usage of the shared resource of available VMs, and succeed in incentivising contribution from the users. This is important in the community cloud model, since without active and continued contribution by the users, the model cannot be sustainable and scalable.

In Chapter 5, we broaden our scope to the overall community network where users participating in the community cloud may be from different zones, and have no prior relationships or established trust among them. We focus specifically on the bandwidth available at the Internet gateways in the community network for two main reasons. Firstly, the available bandwidth to Internet is limited and already in high demand by the users of the community networks, and secondly, the community cloud services will also require access to Internet often with service-level guarantees for the bandwidth resource. We propose a framework for distributed auctioneer that arbitrates users' access to the bandwidth available from the different getaway providers in a fair manner, even in the absence of trust among the users. Even though in Chapter 5 we develop and evaluate the distributed auctioneer for bandwidth allocation, the proposed framework can be generalised and also directly applied to allocating VMs at a cloud service provider. We can also extend this framework for the scenario of trading VMs from Chapter 4, which we leave for the future work.

3.4 Summary

The idea of community network clouds follows on from volunteer computing paradigm. We looked at the general ideas of community clouds, both in commercial and non-commercial context, and focused on the realisation of citizen community clouds within the community networks. Community network clouds are as much a social construct as a technical one, and so require careful design of resource regulation and allocation mechanisms, in order to encourage participation by the members of the community networks.

Notes

The background studies presented in this chapter (§ 3.1) were accomplished in cooperation with my advisors Felix Freitag and Luís Rodrigues. We are also thankful for the cooperation of Roger Baig and Roger Pueyo Centelles from Guifi.net, and Leandro Navarro from Universitat Politècnica de Catalunya (UPC), who provided valuable insights into the operation of Guifi.net community network. The studies were published as a book chapter "*Community Clouds*" [KFN16], in Encyclopedia of Cloud Computing (2016),

published by Wiley & IEEE (ISBN: 978-1-118-82197-8), and as a short paper "*Current Trends and Future Directions in Community Edge Clouds*" [KFR15], in 4th International Conference on Cloud Networking (CloudNet 2015), Niagara Falls, Canada, October 2015.

The work on the proposal for distributed architecture presented in this chapter (§ 3.2) was accomplished in cooperation with my advisor Felix Freitag. We also collaborated with Mennan Selimi and Emmanouil Dimogerontakis, other PhD students at UPC and IST, who did provide relevant contributions for the evaluation of cloud services in community networks. The proposal was published as a full paper "*Towards Distributed Architecture for Collaborative Cloud Services in Community Networks*" [KSF14], in 6th International Conference on Intelligent Networking and Collaborative Systems (INCoS 2014), Salerno, Italy, September 2014. It was later integrated into an extended journal article "*Cloud services in the Guifi.net community network*" [Sel+15], in Computer Networks 93.P2 (Dec. 2015).

4

Managing Incentives with Trusted Users

Community networks are an ecosystem which is able to regulate and maintain itself, some of the community networks are there for even more than a decade. We argue that community cloud, a cloud infrastructure formed by community-owned computing and communication resources, has many technical and social challenges so that the main drivers of today's contribution to community networks, volunteer and altruistic behaviour, are not enough to successfully overcome these barriers. Our hypothesis is that for community cloud to happen, the members' technical and human contribution needed for such a cloud, needs to be steered by incentive mechanisms that pay back the users' contribution with a better quality of experience for them.

In this chapter, we study an incentive mechanism for clouds in community networks, keeping in view the key characteristics of community networks and the scenarios we foresee for community clouds. This incentive mechanism is inspired by Parecon economic model [Alb04; Rah+10; Veg+13], and is based on the idea of effort of each participant, which we define as her contribution relative to her capacity. Our approach is to do the evaluation with simulation experiments and a prototype, which allows us to derive additional conclusions regarding its feasibility for implementation and deployment on a wider scale.

4.1 Motivations

Community networks are based on the principle of reciprocal sharing, which is formalised by the agreements users accept when they join the network. The Wireless Commons License [Wir10] or Pico Peering Agreement [Pic05] is adopted by many community networks to regulate network sharing. The underlying principle is that users allow all the traffic to transit over the links that they own, and this is in return for the access they enjoy to the community network. The agreements are enforced, in most cases, through the social context. For example, members of Guifi.net have mailing lists and many regularly hold meetings where any complaints and issues are also addressed [Bai+15b]. In the worst situation, users may be excluded from using the services, if their behaviour continues to be damaging to the operation of the community network.

Community networks in most cases tend to avoid having sophisticated mechanisms in place like auditing, billing, or other advanced pricing or auction-based mechanism, since they find it not in line with the sharing and open spirit of the community networks. Moreover, such mechanisms introduce additional overheads to the operation of the network, and in many cases may be difficult for users to understand and may discourage participation. When problems do occur, these are often addressed in ad-hoc manner and at the local level, for example, the network administrator may shut down some of the links. For persistent problems, the community often chooses to increase the capacity by buying more equipment and adding new links. For instance, in Guifi.net, some nodes have been successfully crowd-funded [Bai+15b] if such a node was needed by several people. Crowd-funding of a node happened when for a group of people an infrastructure improvement was necessary. For example, an isolated zone of Guifi.net established a super node to connect to other zones.

We think that such a situation may not be sufficient for sustaining a community cloud ecosystem. This is because the contribution required from the users of community cloud, both in terms of the equipment, capital investment, and maintenance costs, and their effort, knowledge, and time, is orders of magnitude larger than what is needed to keep the community networks running. In the absence of any resource regulation mechanism, users will lack incentives to contribute resources. When all the computing and storage are made available at no cost to the community, it would be difficult to find many users willing to contribute solely for altruistic reasons, and even when the resources are made available, they will get consumed quickly by the free-riding users. Therefore, we propose that incentives based resource regulation has to play a key part in the sustainability of a community cloud ecosystem.

4.2 System Model

4.2.1 Nodes in Community Network

We consider a community network, which is managed and owned by the community, and the nodes are managed independently by their owners. The nodes in a community network vary widely in their capacity, function and capability, so we generally divide the nodes into two categories: super nodes (SNs), and client or ordinary nodes (ONs). SNs have multiple wireless links and connect with other SNs to form the backbone of the community network, and are usually intended to be stable with permanent connectivity. Most SNs are installed in the community network participant's premises. A few SNs, however are placed strategically in a third party location, e.g. telecommunication installations of municipalities, to improve the community network's backbone. ONs, on the other hand, are only connected to the access point of a SN. Topological analysis of approximately 17,000 nodes of the Guifi.net community network indicates that around 7% are SNs while the others are ONs [Veg+12].

Principally the hardware for computation and storage is already available in community networks, consisting of some servers attached to the networking nodes. No cloud services, however, are yet deployed in community networks to use this hardware as a cloud, leaving the community network services significantly behind the current standard of the Internet. Our vision is that some community wireless routers will have cloud resources attached, building the infrastructure for a community cloud formed by several cloud resources attached to the nodes. We note that ONs could principally also contribute cloud resources.

On the management side, community networks like Guifi.net, are organised into zones. A zone can be a village, a small city, a region, or a district of a larger city. The organisation of the group within a zone is of many types. Mostly the interests, available time and education of the people drive what happens in the zone. We note that while the allocation of IP addresses and layer 3 networking is agreed among all Guifi.net zones, as it is needed to make the IP network work, the detailed technical support is rather given within the local community of the zone. Therefore, we identify a zone to have the highest social strength within the community network.

4.2.2 Community Cloud Scenarios

These observed characteristics of the community networks lead to two cloud scenarios in our model, local and federated community clouds. The SNs and ONs provide the networking infrastructure, which in order

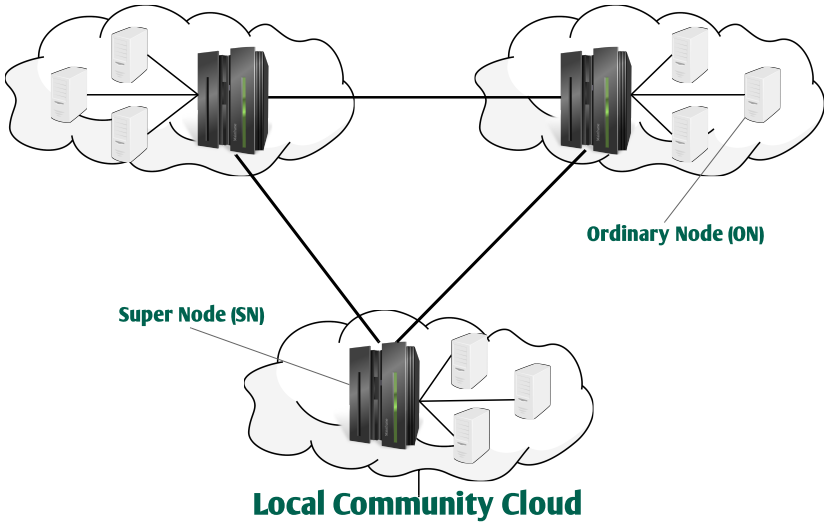


Figure 4.1: Nodes in federated community cloud

to realise the community cloud, need to be attached to other hardware like servers and storage. The cloud management software will run on these servers attached to the networking nodes, SNs and ONs. In the following, when we refer to these nodes, SNs or ONs, we presume them to be those networking nodes of the community networks that already have other computing hardware attached.

First, we have the "local community cloud" scenario, which is derived from the topology of the community network, given by the fact that the community network generally has two different types of nodes, SNs and ONs, and the observed characteristics of the strength of social network within zones [Veg+12]. In such a local community cloud, a SN is responsible for the management of a set of attached nodes contributing cloud resources. From the perspective of the attached nodes, this SN acts as a centralised unit to manage the cloud services.

Next, we consider the scenario when multiple SNs from different zones in a community network connect to form a "federated community cloud". SNs connect physically with other SNs through wireless links and logically in an overlay network to other SNs that manage local clouds. SNs coordinate among themselves for provisioning infrastructure service so the requests originating from one SN's zone can be satisfied by the resources allocated from another SN's zone. Figure 4.1 shows an example of a federated community cloud formed by SNs from three zones. The ONs in a given zone are directly managed by the SN in that zone but they can also consume resources from other zones because of the coordination among SNs.

4.2.3 Resource Provisioning and Coordination

With these two cloud scenarios, we assume that requests for the resources are generated by the users at the ONs. Users, who want to contribute resources also make them available at ONs. All the requests are submitted directly to the single SN managing that particular ON. SN keeps track of all the ONs, the available resources, and the requests from the users. SN services the requests depending on the availability of the resources, and whether the user has sufficient credit to pay for these resources. When a user consumes resources at another ON, the connection is set up through the SN, and all the interactions remain within the local community cloud. When SN decides to federate resources with other zones of community clouds, all the interactions are through the SNs of both the zones.

4.3 Effort-Based Incentive Mechanism

To ensure sustainability and growth of the community cloud, incentive mechanisms are needed that encourage members to contribute with their hardware, effort and time. When designing such mechanisms, the heterogeneity of the nodes and communication links has to be considered since each member brings in a widely varying set of resources and physical capacity to the system. Most peer-to-peer (P2P) systems implement incentive mechanisms based on contribution, where nodes are rewarded according to resources they donate to the system [She+10]. We suggest an effort-based incentive mechanism for community cloud, where effort is defined as contribution relative to the capacity of a node [KBF15]. This mechanism is inspired by the *Parecon* economic model [Alb04; Rah+10; Veg+13], which focuses on social welfare by considering inequality among nodes. Nodes with different capacity cannot have same contribution to the system, but with effort-based mechanism they get same reward if they share as much as possible of their capacity, as we explain in § 4.3.1. We can use a system of credits, acting as virtual currency, to facilitate transactions between providers and consumers. When resources are consumed, providers earn credits which they can later use to request resources from the system. We assume, because of the trust among the community, that users truthfully report the capacities of their nodes.

4.3.1 Formulations

We first discuss the criteria that a SN uses to evaluate requests from ONs. When a node asks for a resource from a SN, which in this case means to commit an instance of VM for a given duration, the SN first checks

whether the ON's credit is sufficient to cover the cost of the transaction. The cost given in eq. (4.1) is proportional to the number of resources requested R_i and the duration T_i for how long they are required. The nonzero coefficients γ and ρ apply to the amount and duration of resources shared respectively, and provide a way to tweak the value generated by the transactions, which is useful to control the behaviour when implementing the prototype system.

$$\textit{transaction\_cost} = \gamma R_i \times \rho T_i \tag{4.1}$$

The cost from all the past transactions node i participates in, either as a provider or a consumer, determines the level of its credit. SN keeps track of the history of all the past transactions, and the credit of each node. If the requesting node does not have enough credit, the request is rejected. Otherwise, the SN searches for nodes that have resources available. It selects as many nodes as possible from its local zone to provide the resources. If the demand cannot be met locally, the SN forwards the request to other SNs in the federated community cloud.

Now we consider how the SN manages the credits of the nodes that take part in the transaction. For each node which contributes its resources to fulfil the request, the SN calculates the transaction cost from eq. (4.1), and adds it to that node's credits. The cost is deducted from the credits of the node that consumed the resources. After the transaction is completed, the effort for each node involved in the transaction is recalculated using eq. (4.2). Here $credit_i$ represents the total credit of the node i, taking into account the outcome of the most recent transaction. The nonzero coefficient ϵ applies to the capacity of the node, and acts as a normalising factor taking into account the overall capacity in the system. A selfish node not contributing enough has effort $E_i < 1$, and will be at a disadvantage when requesting resources, as in eq. (4.4).

$$E_i = \begin{cases} \frac{credit_i}{\epsilon C_i} & if \quad \frac{credit_i}{\epsilon C_i} < 1 \\ 1 & otherwise \end{cases} \tag{4.2}$$

The effort E_i of a node expresses its relative contribution to the system, since the mechanism considers the capacity C_i of a node as well. This means that a node with low capacity puts in more effort than a node with high capacity even if both of them donate same amount of resources to the system. For total N nodes in the system, the total amount of available resources Ω is the sum of the resources ω_i contributed

by each node $i \in N$ (out of its total capacity C_i).

$$\Omega = \sum_{i \in N} \omega_i \tag{4.3}$$

The maximum resources node i can consume, ΔR_i, depends upon its effort E_i. A node actively contributing to the system has $E_i = 1$, and so can access all the available resources ($\Omega - \omega_i$), but a selfish node with $E_i < 1$ gets penalised with limited access to the resources.

$$\Delta R_i = E_i \times (\Omega - \omega_i) \tag{4.4}$$

The effort-based mechanism implies that the nodes with better resources may find it beneficial to under-report their capacity in some cases. This requires that the mechanism should be carefully designed so as not to penalise high capacity nodes too much. Moreover, within a community of trusted users, we assume that the social context encourage users to be truthful when reporting the capacity to SN, and that the community may employ other mechanisms to verify the capacity of the nodes.

4.3.2 Algorithm for Requests Processing

Algorithm 4.1 explains how a SN handles request from a node in its zone. When SN receives a request, it first calculates that node's allowance ΔR_i to confirm whether it has enough credit to fulfil the request. If not, the request is rejected, otherwise the algorithm calls DECISION function which searches for available resources (lines 1–5). The DECISION function first checks if enough resources are available in the local zone (line 8), and selects the nodes that will provide the resources from its local zone (line 9). If SN cannot satisfy request from its local nodes, it forwards request to one of its neighbouring SNs (lines 16–18). After the provider nodes commit resources, SN calculates cost of the transaction and updates the nodes' credits, deducting credits from the requester and increasing credits of the providers (lines 10–14). The sequence of steps is depicted in Figure 4.2.

If the request from an ON is rejected, ON can submit the request again to SN after some time. In the current implementation, resources are assigned to whoever first requests them as they become available. So this may not be fair in the sense that the time users have been waiting for the resources is not taken into account. This issue can be addressed by using queuing at the SN to keep track of the pending requests. Similarly, a related issue occurs when there may be multiple ONs that the SN can pick as providers for a given

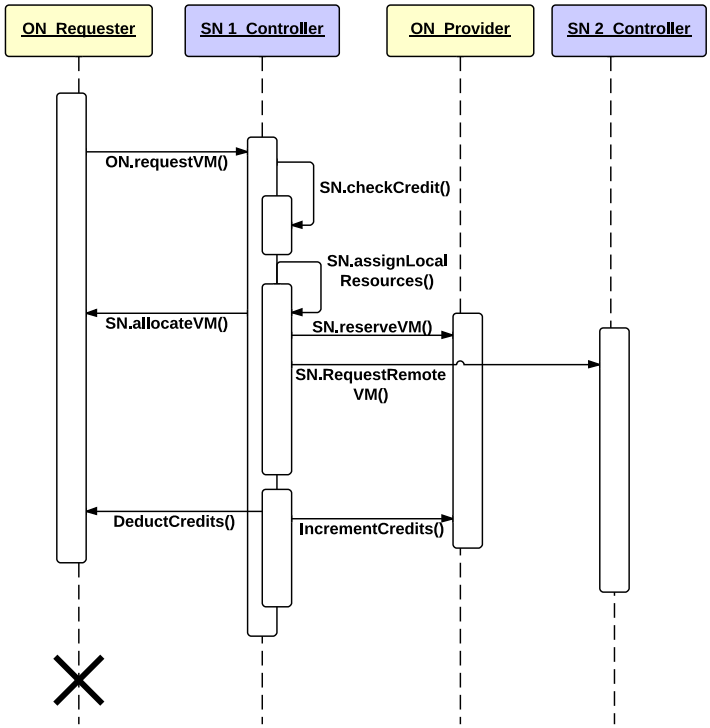


Figure 4.2: Details of the VM request operation by ON

Algorithm 4.1 Handling requests from ONs

Input: receive query from node *i* with the requested amount R_i and the time T_i

calculate (ΔR_i)
if $R_i <= \Delta R_i$ **then**
    call DECISION (i, R_i, T_i)
else
    send ("rejected", i)
end if
procedure DECISION (i, R_i, T_i)
    if $R_i <= \Omega$ **then**
        ProvidersList[n] $\leftarrow$ provider (*ON_List*, R_i)
        for each j in *ProviderList*[n] **do**
            Cost Of Transaction $_{j \to i} \leftarrow R_j^r * T_j^t$
            update_credits (*Cost Of Transaction* $_{j \to i}$)
            update_database (*ON_List*)
        end for
    else
        SN $\leftarrow$ provider (*SN_List*, R_i)
        forward (*SN*, i, R_i, T_i)
    end if
end procedure

request. Therefore, SN may have to apply a selection mechanism for prioritising which ONs to choose as providers. We have studied the effect of some selection mechanisms on the performance in [KBF13], and we leave further investigation of these issues for future work.

4.4 Performance Evaluation

In order to assess the suitability of the proposed effort-based mechanism to community network clouds, we need to study how it impacts the efficiency of the system, and how fair it is to the users with different resource capacities. As compared to contribution-based mechanisms, which reward the users in terms of the total resources they contribute to the system, effort-based incentives may put high contributors at slight disadvantage, when they reward the users with low capacities. So effort-based mechanisms need careful design in order to maintain high system utilisation, and at the same time ensuring fairness for the users with different level of contributions.

In the following, we evaluate the mechanisms first through simulation experiments (§ 4.4.1), and then through a prototype deployed in a testbed with real community network nodes (§ 4.4.2). Our results show that effort-based mechanisms apply well to the context of community network clouds.

4.4.1 Evaluation with Simulation Experiments

We evaluate the impact of the effort-based incentive mechanism through simulation experiments that cover resource regulation on a larger scale across multiple SN zones covering both local and federated community cloud scenarios.

Experiment Setup

We simulate a community network comprising of 1000 nodes which is divided into 100 zones and each zone has one SN and nine ONs. The zones are distributed in a small world topology where each zone is neighbour to 10 other zones. This approximation holds well for real world community networks as, for example, topology analysis of Guifi.net [Veg+12] shows that the ratio of super node to ordinary nodes is approximately 1 to 10. Each ordinary node in the simulation can host a number of VM instances that allows users' applications to run in isolation. Nodes in the zone have two main attributes, one is capacity which is the number of available VM instances, and other is sharing behaviour which is how many

Table 4.1: Configuration for each node in a zone with shared and total instances

Node Behaviour	Shared	Small capacity	Medium capacity	Large capacity
Selfish	33%	ON1 (1/3)	ON2 (2/6)	ON3 (3/9)
Normal	66%	ON4 (2/3)	ON5 (4/6)	ON6 (6/9)
Altruistic	100%	ON7 (3/3)	ON8 (6/6)	ON9 (9/9)

instances are shared with other nodes. Table 4.1 shows the different configurations for each of the nine ONs in each zone. Nodes with low, medium and high capacity host 3, 6 and 9 VM instances respectively and they exhibit selfish, normal or altruistic behaviour sharing one-third, two-thirds or all of their VM instances. For example, node ON2 has medium capacity with 6 instances and exhibits selfish behaviour reserving 4 instances for itself and contributing only 2 to the system.

When the experiment runs, the nodes are given initial credit in proportion to their capacity. Nodes make requests for resources proportional to their capacity asking for two-thirds of their capacity. For instance nodes with capacity of 3, 6 and 9 VM instances request 2, 4 and 6 instances respectively. Nodes request instances for fixed duration and after transaction is complete wait briefly before making further requests. We have implemented the simulator in Python.

Ratio of Successful Requests

Table 4.2 shows the success ratio for requests made by different nodes analysed both with the effort-based and contribution-based incentive mechanisms. We first notice that the success ratio values decrease as the capacity of the nodes increases. This is explained by the fact that nodes with greater capacity request more instances, and so have a higher chance of getting rejected either because there are not many resources available in the system or because the requesting nodes do not have sufficient credit. However, when comparing the success ratio for nodes as their capacity increases, we observe that there is not a great variation. For instance, for the normal sharing behaviour the values range from 66% to 97% for contribution-based incentives, but from 86% to 90% for effort-based incentives. This is explained by the fact that contribution-based approach does not take heterogeneity of nodes into account, and penalises nodes with low capacity as they cannot contribute as much to the system as others. These results indicate that effort-based incentives ensure *fairness* in the system, since the nodes with the same sharing behaviour are treated equally irrespective of their capacity.

Table 4.2: Success ratio for nodes with different configurations (effort vs contribution)

Node Behaviour	Incentives	Small capacity	Medium capacity	Large capacity
Selfish	effort-based	54%	53%	50%
	contribution-based	66%	59%	39%
Normal	effort-based	90%	91%	86%
	contribution-based	97%	77%	66%
Altruistic	effort-based	97%	94%	86%
	contribution-based	97%	85%	65%

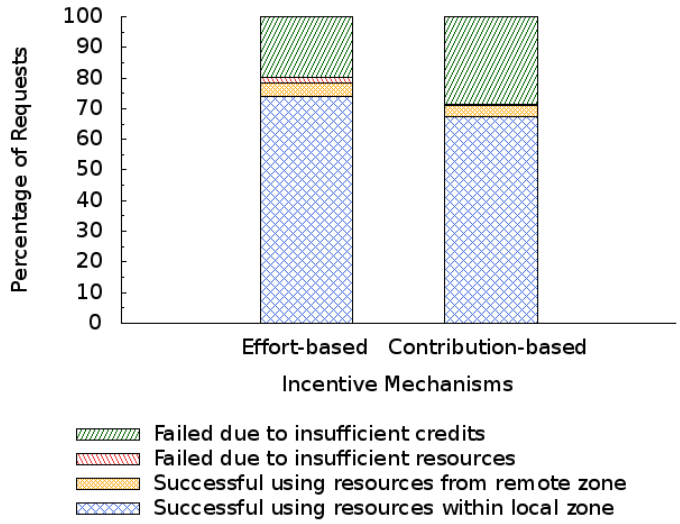


Figure 4.3: Breakdown of outcome of requests with effort and contribution based mechanisms

Breakdown of Request Responses

Figure 4.3 shows the overall breakdown of successful and rejected requests across all the zones, where there are many more successful requests than rejected ones. The success ratio is slightly better for effort-based incentives. Moreover, contribution-based mechanism has greater share of requests rejected because of lack of credit, while very few requests are rejected because of a lack of resources. This indicates that the effort-based incentive mechanism improves efficiency as more resources are utilised. In addition to this observation, the majority of the requests are fulfilled using resources from the local zone with very few requests forwarded to other zones.

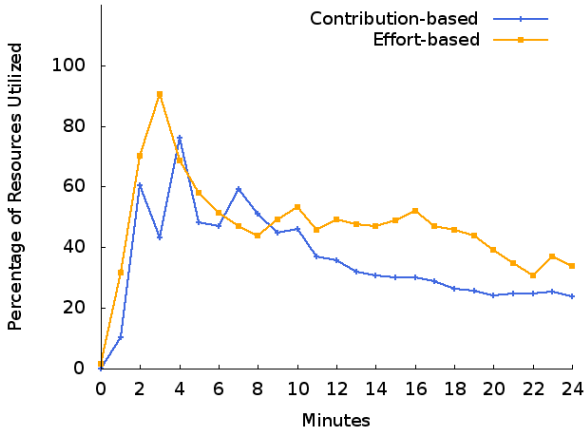


Figure 4.4: Resource utilisation

Resource Utilisation

Figure 4.4 shows the proportion of resources utilised in the system along the execution of a 24 minutes experiment for effort and contribution based approaches. In the beginning, all nodes have enough credit and the resource utilisation is high. Then it drops to below 60% at around the 12th minute, and keeps fluctuating for a while. Afterwards, since most of the nodes have completed their transactions and consumed their credits, the utilisation decreases significantly. The effort-based approach though achieves a higher resource utilisation during that time.

The results also point to a possible load imbalance issue since as the experiment progresses the large capacity nodes may accumulate the credit, lowering the percentage of used resources over time. Many nodes are left with insufficient credit to consume resources. One approach to overcome this issue is to supply all the nodes with a limited fixed amount of credit at regular intervals, which will keep the resource utilisation high.

4.4.2 Evaluation with Prototype

Prototype Implementation

We have implemented a prototype of the incentive-based regulation mechanism [KBF14] in Python using CouchDB [ALS10] database at the back-end, and deployed it in Community-Lab testbed [Com16b]. We chose Python because the current host operating system installed on ONs in the testbed is Open-

WRT [Ope16c], which supports Python, but does not support many other languages such as Java. We selected CouchDB because among its advantages, it is lock-free, schema-less and provides a REST interface, and is also part of the other components of the SN's cloud management software being developed. In the SNs, Debian operating system is installed.

Figure 4.5 shows the implemented components of the cloud coordinator from the architecture in § 3.2. The components from the prototype are described below.

ON Management ONs can register with SN to request and to contribute resources.

Regulation Mechanism When pooling resources from multiple zones, the cloud coordinator applies a regulation mechanism that takes into account resource utilisation and contribution by different nodes to perform resource allocation.

SN Interconnectivity The design of a community cloud manager follows a decentralised approach, so cloud coordinator relies on gossip-based discovery mechanisms to manage overlay network of the SNs in community cloud. The updated list of adjacent SNs is saved in SN-List database.

SN Resource Sharing When requests from ONs cannot be met from resources in the local zone, SN can request resources from other SNs in the system.

ONs use the remote procedure call (RPC) mechanism to connect to the SN. First of all, an ON assigns itself to a parent SN with a register message which includes metadata of that ON such as IP address, total capacity and number of VMs shared. In the current prototype, the IP address of SN is fixed and is manually provided when setting up ONs. However in future, service discovery tools like Serf [Ser16] can be used to get address of SN. This registration information is stored in the ON-List database of the parent SN by creating an entry for the corresponding ON. After that, the ON is ready to send requests to its parent SN, which are processed using Algorithm 4.1 as shown in Figure 4.2. When an ON requests its parent SN for any VMs, it specifies the duration for how long it needs to use the VMs. This request is evaluated by performing incentive and decision mechanisms as explained in section 4.3. If a request cannot be met locally, the corresponding parent SN checks its SN-List database to find another zone with available resources. The interactions between SNs are also made through RPC mechanism. In the SN controller software, there is a separate process which regularly checks the database for any updates. If the

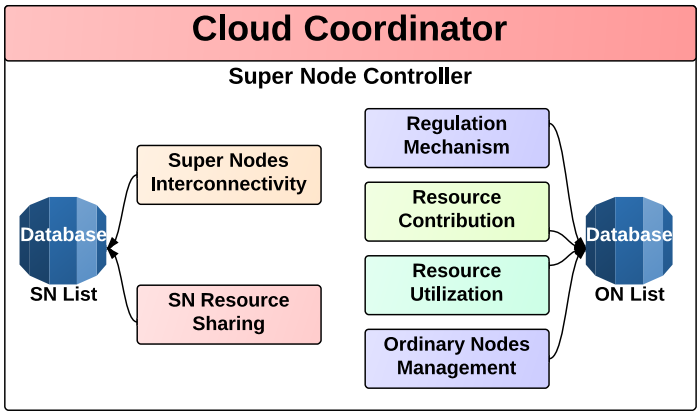


Figure 4.5: Components of cloud coordinator

duration of a consumer ON's resource request has expired, it frees the VMs and makes them available for the provider ON, and updates the metadata entries of the corresponding ONs in the ON-List database. The current implementation keeps track of the number of VMs contributed and consumed by each ON. The system copes with ONs connecting and disconnecting from the SN at any time since ONs periodically send heartbeat messages to the SN. The design allows us to include values of metrics like CPU, memory, and bandwidth usage in the future for fine-grained decisions about resources assignment.

Experiment Setup

We deploy the prototype of the regulation component of the cloud coordinator from community cloud management system in the Community-Lab testbed, which is developed by the CONFINE European project [Bra+13]. The cloud coordinator components are installed on nodes of the Community-Lab testbed, which consist of Jetway JBC372F36W devices, and are equipped with an Intel Atom N2600 CPU, 4GB of RAM and 120GB SSD. Depending on the experiment, one or two nodes operate as SNs, while each ON hosts between one and four VM instances. Note, however, that since OpenWRT has limited supported for either containers-based or full virtualization, in these experiments the nodes submit and process the requests but VMs are not actually created on the ONs. The objectives of the experiments are twofold:

1. Experiment 1: Assess the prototype operation regarding the incentive-based resource assignment algorithm in a local community cloud scenario.

2. Experiment 2: Study the coordination between SNs from different zones in the federated com-

munity cloud scenario with heterogeneous resource distribution.

Resource Assignment in Local Community Cloud Scenario

In order to study the performance of the prototype in a real deployment of a local community cloud, we install our software components in four ONs and one SN in Community-Lab testbed, which are connected to the Guifi.net community network. Each node behaves as an ON but with different configuration, in order to have a heterogeneous set of cloud resources. The four nodes include f101 sharing 1 out of total 2 VMs, f102 sharing 3 out of total 3 VMs, f103 sharing 1 out of total 3 VMs, and f104 sharing 1 out of total 1 VM. Each ON sends request for VM instances to the SN at regular intervals. VMs are requested for 20 seconds interval at a time. Each ON requests as many VMs as its total capacity, for example node f101 always requests 2 VMs. If the request is accepted by the SN, the ON obtains the VMs for the next 20 seconds. If the request is rejected, the ON waits for 5 seconds before making any further requests. The experiment is run with this setup for around 5 minutes. We analyse the different aspects of the system behaviour in the following.

Resource Utilisation

Figure 4.6 shows the level of resource utilisation in the system in terms of the number of reserved VMs versus the total number of VMs. It can be seen that resource utilisation varies widely and 100% utilisation, meaning all the VMs being occupied, occurs only for short intervals. This is because as nodes obtain VMs they spend their credit, and if they are not able to earn credit by contributing VMs, their credit gets below a certain threshold, and they can no longer request more VMs. At approximately 80th second, the utilisation gets very low. Nodes then need to earn credit by providing VMs to others before they can request VMs again. So even though VMs are available, they cannot be utilised due to the lack of credit in the system.

Credit Distribution

Figure 4.7 shows the credit distribution among the four ONs during the 5 minutes of the experiment. A node's credit is affected by how many VMs it shares and how much credit it spends to obtain VMs. When a node shares most of its capacity, like ON f102 providing all its 3 VMs, it earns more credit and so maintains a high credit level during the experiment. On the other hand, when a node continuously consumes VMs

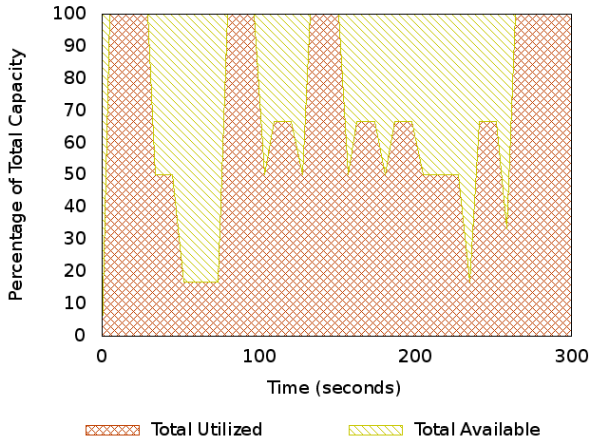


Figure 4.6: Overall resource utilisation of the four ONs

like ON f101 and f104, it keeps on spending any credit that it earns, so its credit does not increase beyond a certain level. Of particular interest is the behaviour of ON f103, which earns credit in the start and gets a spike in credit level halfway through the experiment, but then quickly spends it as it requests VMs from others.

Note that an ON's credit can be negative or higher than 100% of the total credit because in the current implementation SN can allow requests from ONs with zero or or less than zero credit up to some extent. The ONs with zero or negative credit can, of course, continue to provide VMs and earn credit, they can only not request VMs if their credit is negative and below a certain threshold. This allows the nodes without any credit an opportunity to continue participating in the system and increase their credit by contributing resources.

Success Ratio

Figure 4.8 shows the ratio of the fulfilled requests for each node, which is affected by the level of credit of the node and the amount of resources available in the system. ON f104 has the most success since it requests only one VM at a time while ON f103 has the least success since it requests 3 VMs, which is half of the total shared VMs in the system. ON f101, on the other hand, gets its requests rejected because of the lack of credit. Therefore, this node has to wait to gain the needed credit.

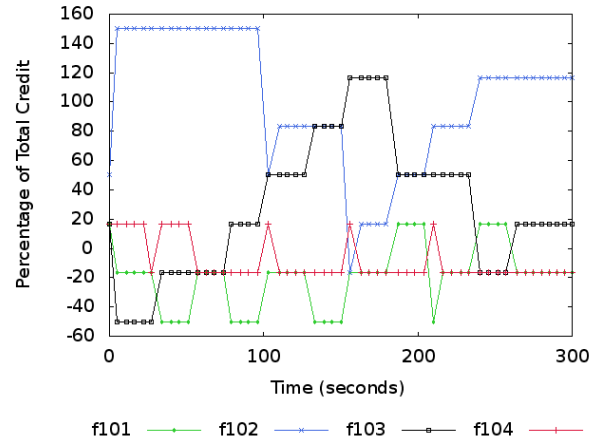


Figure 4.7: Distribution of credit among the four ONs

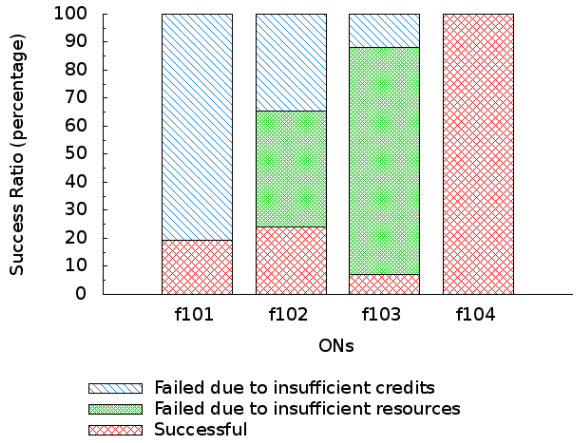


Figure 4.8: Ratio of fulfilled and rejected requests

Resource Assignment in Federated Community Cloud Scenario

In this experiment, we set up two local clouds, each with one SN and four ONs to study the federated community cloud scenario, as illustrated in Figure 4.1. Table 4.3 shows the two cases with different number of VMs available in the two zones. In the case of scarce capacity (case 1), the nodes in the SN1 zone share very few VMs compared to nodes in SN2 zone. In the case of equal capacity (case 2), the nodes in both the zones share the same number of VMs.

Figure 4.9 shows the proportion of the requests fulfilled by VMs provided by the other zone. With scarce capacity in SN1 zone, around 50% of the requests are fulfilled by VMs provided by SN2 zone. SN2

Table 4.3: Two cases with different resource distribution between zones

		Case 1: Scarce Capacity		Case 2: Equal Capacity	
SNs	ONs	Total VMs	Shared VMs	Total VMs	Shared VMs
SN_1	ON_1	3	1	3	2
	ON_2	3	1	3	3
	ON_3	3	1	3	2
	ON_4	1	1	1	1
SN_2	ON_1	3	2	3	2
	ON_2	3	3	3	3
	ON_3	3	2	3	2
	ON_4	1	1	1	1

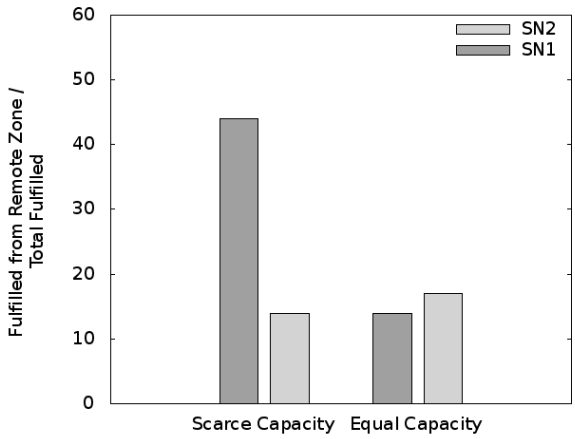


Figure 4.9: Resources assigned from different SN zones

with sufficient capacity is able to meet most of the requests from VMs within the same zone, forwarding less than 15% requests to the other zone. In the second case, when both zones have the same available capacity, most of the requests get processed within the same zone for both the SNs. This shows that a federated community cloud scenario extends the resources assigned to zones with limited capacity.

4.4.3 Discussion

From the simulation experiments (§ 4.4.1) we find that the effort-based resource regulation succeed in not only ensuring high system utilisation, but they also ensure fairness when rewarding users for their contributions. Following on from the insights through the simulation experiments, we implemented a prototype and from the results through its deployment in Guifi.net community network (§ 4.4.2), we

observe that:

1. The prototype of the regulation service deployed in real community network nodes performed the required operations. Its components worked correctly both in the ON's host operating system (OpenWRT) and the SN's operating system (Debian). We could not observe the limitations of our implementation within scales that are realistic for community networks. We note however that as a continuation of this work, a more extensive deployment of several federated community clouds with real users and real usage should ultimately be undertaken.

2. The algorithm used for the regulated resource allocation controlled the VM assignments, taking into account the user's contribution and usage. More complex situations, however, should be created in further studies to provide additional insight into how the system behaves.

3. Our experiments were carried out on limited number of nodes and for limited time. If our prototype was deployed on additional nodes that are geographically widely spread and run for extended periods, the VM assignment decisions might need to take into account information from network awareness to select the appropriate cloud resource providers.

4.5 Summary

We focused in this chapter on the need for an incentive-driven regulation mechanism, and identified it as a key component to encourage contribution towards and foster adoption of community clouds. We investigated incentive mechanisms for community clouds based on reciprocal resource sharing, and the results highlight their impact on the efficiency of the system and on regulating the resource assignments. The regulation component is implemented in a simulator in order to be able to perform assessments for large scale scenarios. With simulation experiments we characterised the behaviour of different settings of the incentive mechanism, and evaluated the success ratio of requests by the nodes and the resource utilisation. We implemented a prototype and deployed it in Guifi.net to study the performance of the proposed mechanism under the real world constraints. We find that effort-based incentive schemes perform well in maintaining a community cloud ecosystem, through ensuring continued contribution by the users. The understanding gained from these results helps in the design of the policies that such incentive mechanism could follow in a future platform of real community cloud system.

Notes

The results presented in this chapter were accomplished in cooperation with my advisor Felix Freitag and Ümit C. Büyükşahin, a Masters student at UPC [Buy13]. Ümit did provide relevant contributions to the implementation of the simulator, and the prototype.

The results were published as full papers "*Towards Incentive-based Resource Assignment and Regulation in Clouds for Community Networks*" [KBF13], in 10th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2013), Zaragoza, Spain, September 2013, and "*Prototyping Incentive-Based Resource Assignment for Clouds in Community Networks*" [KBF14], in 28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014), Victoria, Canada, May 2014. An extended version of the work was then published in the article "*Incentive-based Resource Assignment and Regulation for Collaborative Cloud Services in Community Networks*" [KBF15], in Journal of Computer and System Sciences 81.8 (Dec. 2015).

5

Managing Incentives with Untrusted Users

When cloud applications are deployed within community networks, in many cases connectivity external to the community network is important. Even when cloud applications are deployed using resources solely available within community networks, they require connection to the Internet for functioning. In the basic case, cloud applications may want to backup or synchronise data with servers external to the community network, or require fetching data for operating the service, for instance a video-on-demand service may download fresh content. Also, a service available in multiple community networks requires access at the gateways for exchanging data, and gateways in this case act to federate the community networks. Applications for Internet of Things and smart cities involve collecting data from the sensors, which may have to be shared with servers outside the community network for data analysis. For the case of edge clouds, the servers residing within the community networks, acting as nano data centres, require connectivity to the data centres. In all these situations, the applications deployed on servers within the community network require bandwidth at the gateways with quality-of-service (QoS) guarantees to connect to the Internet, though their requirements for robustness, waiting time, throughput, and prioritising network traffic, may vary for different scenarios. Figure 5.1 shows how such an edge cloud can be deployed within

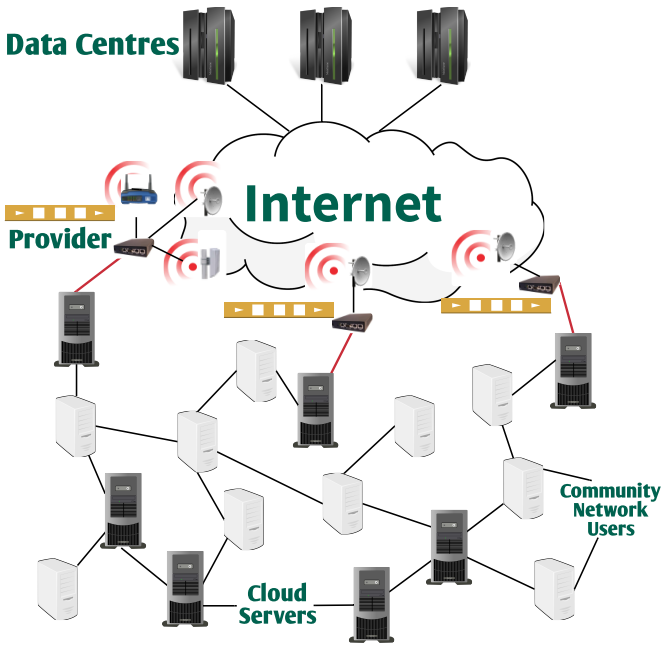


Figure 5.1: Users connected to the service provider's gateway in a community network

a community network. The servers are present at different locations, either caching content for media-rich applications or performing computation locally for time-critical applications. These servers require connection to the data centres through the Internet, for which they rely on the gateway providers available in the community network [Bai+15b].

Community networks, as any other human organisation, are not immune to overuse, free riding, or under-provisioning, specially in scenarios where users may have motivations to compete for scarce resources. In this chapter, we consider the concrete problem of bandwidth reservation on the gateways that connect the community network to the Internet. In particular, we consider that, as in most community networks, the subset of users that offer gateway services to the Internet is smaller than the complete set of users. Users that are not at the gateways may be interested in reserving bandwidth in these gateways for Internet access. If the available bandwidth at each gateway is not enough to satisfy the demand, one needs to implement some arbitration mechanism that optimally and fairly allocate resources, Note that we are not considering the bandwidth on the links internal to the community network, because network is operated in a shared manner, and this bandwidth is collectively owned by the community.

Most of the auction-based game theoretical approaches in literature, as we discussed in § 2.3, assume that the auctioneer is trusted. This is an unreasonable assumption in community networks, where no entity

can be trusted, and even if such entity existed, it would be a bottleneck. Thus, there is a substantial gap that needs to be bridged to apply these results in community networks. In this chapter, our aim is to address this gap by by proposing a framework of distributed protocols that allows multiple resource providers in a community network to simulate the role of the auctioneer. Such simulation raises significant challenges both from the theoretical and practical points of view. From the theoretical perspective, although there is a vast literature of distributed fault-tolerant algorithms, with very few exceptions (for instance, [ADH13]), these works do not consider rational behaviour. From the practical perspective, the distribution of the auctioneer may incur additional overhead. Our proposal addresses both concerns. First, we prove that our distributed simulations are sound from a game theoretical perspective. Specifically, we show that the simulations are k-resilient (ex post) equilibria [ADH13], i.e., Nash equilibria resilient to asynchrony and coalitions of providers of size at most k. Second, our protocols leverage the distributed nature of the resulting virtual trusted entity to parallelise the resource provisioning algorithm, compensating for the additional costs imposed by coordination. We have implemented a prototype of our protocols and evaluated the resulting system in a real community network.

Even though in our example we only focus on bandwidth reservation, the fundamental problems at hand are general and emerge every time shared resources need to be allocated to users in a set of providers, such as for allocation of processing and memory resources as virtual machines in public clouds [Zha+15b], assignment of frequencies in secondary wireless spectrum markets [Zho+08], and resource scheduling in grid and cloud infrastructures[Lai+04].

In summary, our main contributions in this chapter are the following:

- We propose a framework for devising distributed protocols executed among service providers that correctly simulate the auctioneer in a family of resource allocation auctions.
- We show that every implementation of the framework is a k-resilient (ex post) equilibrium. These implementations also tolerate users that send invalid bids.
- We show that it is possible to leverage the distributed nature of our framework to parallelise implementations, mitigating scalability issues of purely centralised solutions.
- We implement instances of the framework and report the results from its deployment on the actual Guifi.net nodes.

The rest of the chapter is organised as follows. First, we discuss the motivation for a truthful and

trusted resource allocation mechanism in § 5.1. Next, we provide the system model in § 5.2. We present the framework for simulating the auctioneer in § 5.3, and we illustrate its applicability to the execution of two different auction mechanisms, with different computational properties, in § 5.3.3. We discuss results from the experiments using a prototype for distributed auctioneer in § 5.4.

5.1 Motivations

We have seen in Chapter 4 how incentives based resource regulation is important to encourage contribution and thus ensure a sustainable community cloud ecosystem. However, the mechanisms in Chapter 4 assume the users will always follow the prescribed polices, and just like in community networks, the social context and the ties between the community would be sufficient to correct any erroneous behaviour. But these assumptions do not always hold true, especially when the community cloud would grow to a large user base. With weakening of any direct social interaction, the system may be open to abuse by the selfish or malicious users. This can negatively impact the viability of community cloud model.

In order to understand better the implications of untruthful behaviour, we study the effects of untruthfulness on the utility obtained by the users and the social welfare of the overall system [Kha+15b]. We first present a model that differentiates between cloud applications with different priority classes. Next, we use this model to evaluate the impact of untruthfulness on the social welfare through simulation experiments, for the different pricing mechanisms proposed in the literature [MT14].

5.1.1 System Model

We consider a bandwidth provider $\mathcal{P}$ in the community network and a set of N users $\{1, 2, ... N\}$. The provider operates a gateway to the Internet, which allows access to resources outside the community network for the users. The users are connected to the provider's gateway through the wireless and fibre links in the community network [Bai+15b], and the applications in community network cloud access their external servers and Internet through this gateway. Figure 5.1 shows the users in the community network connected to the provider through multiple such paths, where only few of these users are the clients of the provider for reserving bandwidth.

The provider processes the requests in a queue at the gateway, where time in the queue is divided into an infinite sequence of slots starting from 1, where all the available bandwidth is allocated to exactly one user in each slot. The provider allocates the slots in batch after receiving all the requests from N users and

assigns the next N slots, one to each user. We choose this model for simplicity, and we divide users into different priority classes based on their slots in the queue, as discussed below. Our findings are applicable to other models, for instance where all the available bandwidth is shared between the users at the same time. In this case, we can have priority classes based on the quality of different links, for example. Considering the provider has two links to the Internet, high priority requests are assigned to the better link with low latency, less packet loss, etc., while the low priority requests are assigned the other link. The discussion and findings presented below can equally apply to this alternate model as well.

We divide the users into two priority classes, $h \in \{0, 1\}$, some have lower priority requests, h_0, and some have higher priority requests, h_1. Here in this model, the main consideration for higher priority requests is that they are more sensitive to the waiting time, and prefer to reserve earlier slots in the queue. Provider $\mathcal{P}$ aims for an optimal schedule when allocating the slots to the users, so as to maximise its revenue and the overall utility for all the users. We provide formal details below.

Schedule A schedule ϕ maps each time slot t to a user i.

Value For any schedule ϕ and user i, let t be the slot assigned to user i, then $v_i(h, t)$ is the valuation given by i for being allocated time slot t, where $h \in \{0, 1\}$ is the priority class of the user. v_i is communicated by each i to $\mathcal{P}$ beforehand.

Utility For any schedule ϕ which assigns user i a slot t, the utility $u_i(\phi, t)$ for user i is difference between the value v_i and the payment $p_i(\phi, t)$ made by user i to $\mathcal{P}$.

$$u_i(\phi, t) = v_i(h, t) - p_i(h, t) \tag{5.1}$$

Restriction Any slot t can be assigned to at most one user.

Optimization Find ϕ that maximises the social welfare, which is the sum of utilities u_i of all users, while fulfilling the restrictions.

$$\text{maximise } \textit{welfare}\,(\phi) = \sum_{i \in N} u_i(\phi, t) \tag{5.2}$$

Scheduler Function S that maps $\vec{u} = (u_i)_{i \in N}$ to optimal ϕ.

Goal A user i when submitting the request to $\mathcal{P}$, declares the priority class h_i and value v_i, and also the bid amount b_i where applicable. When the user behaves truthfully the reported value v_i^* is the same as her inherent value v_i. We want to ensure that it is in the interest of every i to declare her true value of v_i, regardless of the declared values v_j for any $j \neq i$. Such a mechanism is said to be truthful in dominant strategy, where users have no incentive to misreport their values [NR99].

5.1.2 Pricing Mechanisms

Given the above model, the prices are calculated for the bandwidth usage according to different mechanisms [MT14].

Fixed Pricing In the case of fixed usage-based pricing, all the users pay the identical price c_0 for each unit of bandwidth consumed, which is constant irrespective of the priority class.

Priority Pricing In the case of priority pricing, users pay according to the priority class h. Since in our model, there are only two priority classes h_0 and h_1, provider $\mathcal{P}$ charges two different prices c_{h_0} and c_{h_1} per unit of bandwidth, respectively.

First-Price Auction In the case of sealed first-price auction, users make different bids b_i depending on their priority class h, with high priority requests quoting higher bid amounts in general. Each winning user pays their bid amount.

$$p_i(\phi, t) = b_i \tag{5.3}$$

Generalised Second Price (GSP) Auction In a generalised second price (GSP) auction, users make different bids b_i but in this case the winning user pays the amount corresponding to the next highest bidder [Jan+11]. So the user with the highest bid pays the amount of the second highest bidder, and the second highest bidder pays the amount of the third highest bidder, and so on.

Vickrey-Clarke-Groves (VCG) Auction VCG is a second-price sealed-bid auction based mechanism, which ensures truthfulness and maximum social welfare [NR99], if the provider $\mathcal{P}$ can calculate optimal

schedule ϕ in polynomial time. Each user i provides a bid b_i to $\mathcal{P}$, and given a schedule ϕ, each user i pays the price $p_i(\phi, t)$ according to:

$$p_i(\phi, t) = \sum_{\substack{j!=i \\ j \in N}} \left(v_j(h, \phi') - b_j\right) - \sum_{\substack{j!=i \\ j \in N}} \left(v_j(h, \phi) - b_j\right) \tag{5.4}$$

where ϕ and ϕ' are the schedules that maximise $\sum_{i \in N} u_i$ while including and excluding the bid b_i by user i from the allocation respectively.

5.1.3 Scheduling Algorithm

We consider a simple scheduling algorithm which applies a greedy approach for mapping users' requests to the available slots. Algorithm 5.1 shows the scheduling algorithm, where $\mathcal{P}$ assigns the slots to the users in non-increasing order of their reported bids (and corresponding h_i and v_i) for the bandwidth resource. The prices calculated are dependent on the pricing mechanism, the priority class h of the requests, and the assigned slot t in the schedule ϕ. The runtime of the algorithm is $O(N\,log\,N)$ for N users for the different pricing mechanisms. VCG mechanism, however, requires computing N schedules for calculating payments for the N winning bids, so the running time in the case of VCG is $O(N^2\,log\,N)$.

The greedy approach, in general, does not always provide an optimal allocation, which is a pre-requisite for VCG mechanism. However, in the case of the model given above and considering the step function we are going to use for $v_i(h, t)$ from Figure 5.2, the greedy approach from Algorithm 5.1 always returns an optimal allocation. This can be proven through induction, and can be explained intuitively as follows. Selecting the requests with higher bids first (corresponding to higher h_i and v_i) will always give the maximum social welfare, since the value function in Figure 5.2 is non-increasing with time and choosing a bid with lower amount causes a loss in social welfare which cannot be recovered as the time progresses.

5.1.4 Evaluation

We conduct the simulation experiments using the multi-agent programmable modelling environment NetLogo [Wil99]. In all the experiments, we consider a single provider and 500 users. We run the experiments for 1000 rounds, and plot the average values in the graphs.

For different pricing mechanisms (as explained in § 5.1.2), we use the following values. For fixed pricing, we set $c_0 = 0.5$. For priority pricing, we set $c_{h_0} = 0.25$ and $c_{h_1} = 0.75$. For auctions based

Algorithm 5.1 Scheduling algorithm for ϕ, allocating $\vec{t}$ slots to N users

Input: List of users $\vec{n}$, bids $\vec{b}$, for total N users
Output: List of assigned slots $\vec{t}$, and payments $\vec{p}$
1: Sort $\vec{n}$ users in non-increasing order on their bids $\vec{b}$
2: **for** $i = 1, ... N$ **do**
3: $\quad \vec{t}[i] \leftarrow \vec{n}[i]$ ▷ Assign slots
4: **end for**
5: **for** $i = 1, ... N$ **do**
6: $\quad \vec{p}[i] \leftarrow$ *payment* $(\vec{b}[i], \vec{t}[i])$ ▷ Calculate payments
7: **end for**

pricing, the bids for lower priority requests h_0 are uniformly distributed in the range $[0.25, 0.5]$, while the bids for higher priority requests h_1 are uniformly distributed in the range $(0.5, 0.75]$. For differentiating between the two priority classes, we choose different time-utility functions (TUF), which in this case we have chosen as step functions for simplicity. According to this step function, the value $v_i(h, t)$, based on priority class h and slot t in schedule ϕ, decreases for both higher and lower priority classes after a threshold $t_0 = \frac{N}{2}$, as shown in Figure 5.2. Specifically, for lower priority class h_0:

$$v_i(h_0, t) = \begin{cases} 1.5 & \text{if } t \leq \frac{N}{2} \\ 1 & \text{if } \frac{N}{2} < t \leq N \end{cases} \tag{5.5}$$

And for high priority class h_1:

$$v_i(h_1, t) = \begin{cases} 3 & \text{if } t \leq \frac{N}{2} \\ 2 & \text{if } \frac{N}{2} < t \leq N \end{cases} \tag{5.6}$$

Each user i submits exactly one request to $\mathcal{P}$, declaring her priority class h_i, value v_i, and bid amount b_i where applicable. Both the priority classes, h_0 and h_1 occur with the same probability, so almost half of the requests are of higher priority, and the rest are of lower priority. We model lying behaviour of the users, by randomly flipping their reported priority class h to $\mathcal{P}$, according to a uniform distribution. When the users lie, we observe the normalised difference from the case where all the users are truthful.

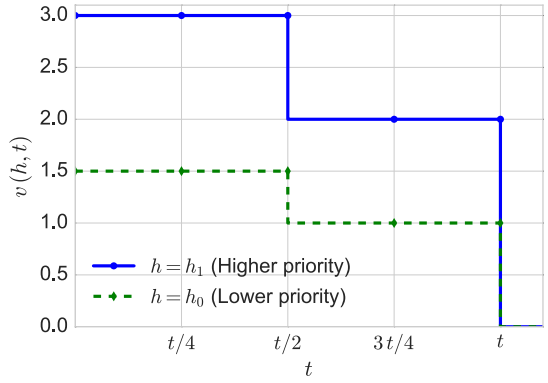


Figure 5.2: Value function $v_i(h,t)$ for user i based on priority class h and slot t in schedule

Here, $u_i^*(\phi,t)$ indicates the case where the users lie to $\mathcal{P}$, and $u_i(\phi,t)$ where all the users are truthful.

$$\Delta \textit{ welfare} = \frac{\sum_{i \in N} u_i^*(\phi,t) - \sum_{i \in N} u_i(\phi,t)}{\sum_{i \in N} u_i(\phi,t)} \tag{5.7}$$

Social Welfare

Figure 5.3 shows how social welfare is affected when the probability $p\,(lying)$ of a user misreporting her value to $\mathcal{P}$ increases up to the point where 90% of the users may be lying. As expected, social welfare decreases as the probability of lying increases, since $\mathcal{P}$ fails to allocate better slots for higher priority requests. All the pricing schemes behave similarly as the proportion of lying users increases, except VCG which performs marginally better in that social welfare is slightly higher for VCG as compared to the other schemes. This shows the importance of encouraging truthful behaviour in the users for maximising social welfare.

Individual Gain in Utility for Different Classes

To understand how the pricing mechanisms incentivise truthfulness for different priority classes, in the next experiment we look at the normalised difference in utility for an individual user (on average), separately for h_0 and h_1. Here again, $u_i^*(\phi,t)$ is the individual utility when some of the users lie, and $u_i(\phi,t)$

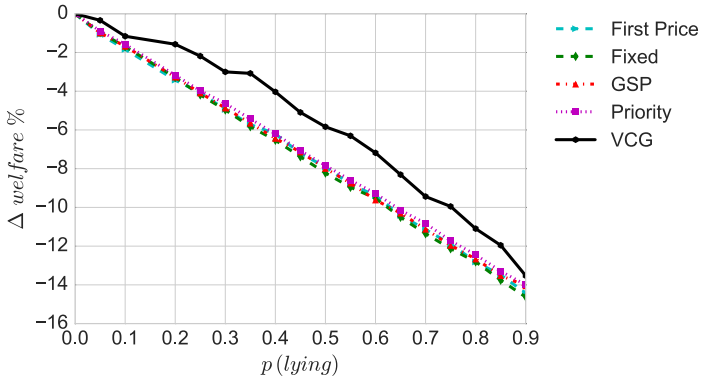


Figure 5.3: Percentage difference in social welfare as more users lie

is when all the users are truthful.

$$\Delta \textit{ utility} = \frac{u_i^*(\phi, t) - u_i(\phi, t)}{u_i(\phi, t)} \tag{5.8}$$

Figure 5.4 shows the percentage difference in the average utility for all the users with low priority requests. Note that this average is over all the users in h_0, and not only those who lie. Users from h_0 may lie in order to get higher value (through reserving an earlier slot), hoping to still pay as little as possible. Figure 5.4 shows that for fixed usage-based price, they do gain in utility since they are paying the same amount for a better service. For priority pricing, they gain nothing as any gains in utility are offset by the higher price. For first price and GSP auctions, the results are similar and there are gains due to lying, though less than those in the case of the fixed price. The first price and GSP auctions behave similarly since expected payments are the same in the first and second price auctions, when the bids are independent and identically distributed [MT14], as is the case in this experiment. VCG performs better since the utility decreases as more users lie.

Figure 5.5 shows the percentage difference in the average utility for all the users with high priority requests. Note that this average is over all the users in h_1, and not only those who lie. Users from h_1 may lie in order to save on their payments, with the hope that they can still get the same value (through keeping their earlier slot). Figure 5.4 shows that users from h_1, in general, lose by lying since there is little chance that $\mathcal{P}$ will assign earlier slots to the users declaring low priority to $\mathcal{P}$. So even though they save on the payments, the decrease in value because of getting assigned later slots results in net loss for users from h_1.

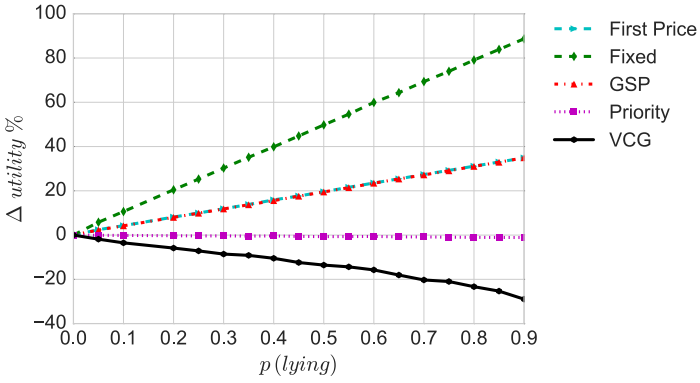


Figure 5.4: Percentage difference in utility for low priority class h_0

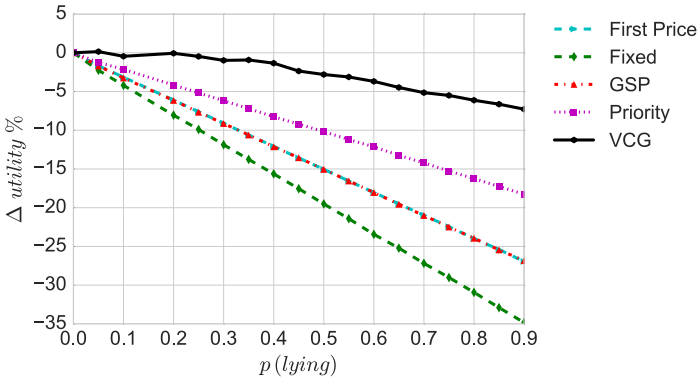


Figure 5.5: Percentage difference in utility for high priority class h_1

Maximum Gains

In the next experiment, we look specifically at the utility for the users that report untruthful values to $\mathcal{P}$, to see the maximum gain they can get in the utility under different pricing mechanisms. Figure 5.6 shows the maximum gain in utility a user from h_0 can get as the number of lying users increases. Note that in this case we pick only the maximum utility for a user from h_0 that is lying, averaged across all the experiment runs. The results are similar to what we observed earlier in Figure 5.4.

Similarly, Figure 5.7 shows the maximum gain in utility a user from h_1 can obtain through lying. We noticed in Figure 5.5 that on average the users from h_1 do not gain through lying, but here we see that for all the pricing mechanisms except VCG, the utility for a lying user with high priority request increases with increase in the number of lying users, though the net gain is not significant. For VCG, the number of lying users does not have much impact, and the loss in utility for the lying user remains almost the same.

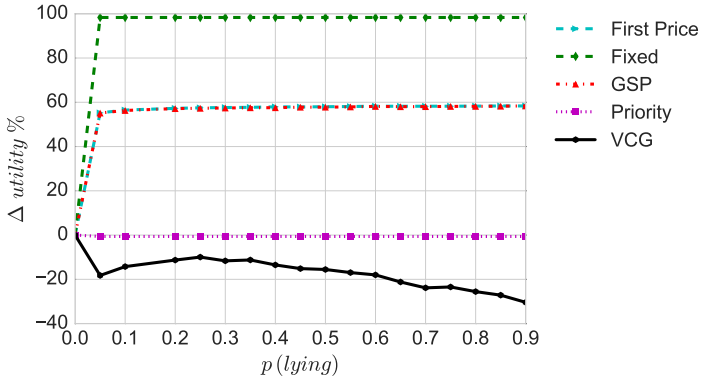


Figure 5.6: Maximum gain in utility for a user from low priority class h_0

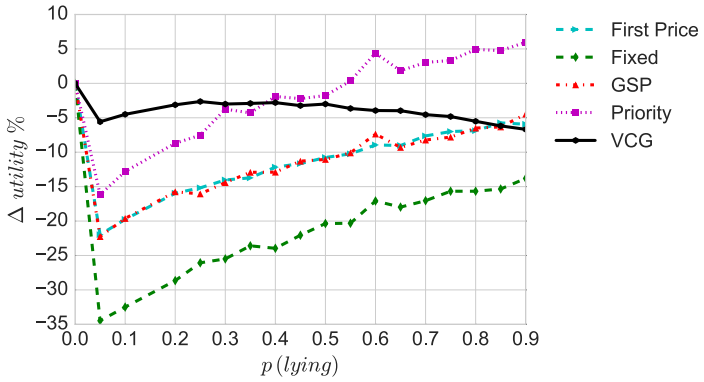


Figure 5.7: Maximum gain in utility for a user from high priority class h_1

Moreover, first price and GSP auctions perform better than priority pricing here. The user can have a net gain in utility by misreporting her priority class as h_0 when more than half of the users are lying in the case of priority pricing.

5.1.5 Discussion

We find that static pricing schemes like fixed usage-based pricing and priority pricing are not very useful for arbitration between requests from different priority classes, since it is hard to avoid everyone reporting their requests as higher priority [MT14]. Dynamic pricing, for example, based on first-price auction can help here but with this simple auction scheme users report bid amounts lower than their true valuation of the bandwidth resource [MT14]. VCG mechanism, when either using optimal allocation algorithms [NR99] or approximate allocation algorithms [Zha+15b], can ensure truthfulness but is often

computationally intensive to implement in practice. Generalised second price (GSP) auction mechanism, an extension of VCG, is not as computationally intensive as VCG and even though it doesn't guarantee truthfulness, it shares many desirable properties of VCG [MT14].

These results show that auction-based mechanisms are good candidates for using in allocation algorithms for bandwidth reservation in community network clouds. However, these mechanisms assume that a centralised auctioneer exists that can be trusted to to execute the allocation algorithm as designed. In the absence of a trusted auctioneer, the truthfulness guarantees of mechanisms like GSP and VCG no longer hold. To address the shortcoming that a centralised trusted auctioneer is not feasible in community network clouds, we present our proposal for a virtual distributed auctioneer in § 5.3.

5.2 System Model

We define the family of resource allocation auctions, the requirements of a distributed simulation of the auctioneer, and the Game Theoretical model used to analyse simulations. These provide basis for our proposed framework for distributed auctioneer (§ 5.3).

5.2.1 Resource Allocation Auctions

Consider a family of auctions with m *providers*, n *users*, and an *auctioneer*. Providers sell multiple resources with a limited capacity, in exchange for payments in some currency. Users are willing to pay to the providers in exchange for the allocation of a minimum amount of each resource in the provider. The auctioneer defines an allocation between users and providers that is *feasible* (i.e., that does not exceed the capacity of each resource in any provider) and defines the payments to be made/received by the users/-providers, respectively. Both users and providers attribute a utility to each allocation, which is a function of the value given to the allocation and the payments made/received. More precisely, each user i has a valuation v_i specifying how much i is willing to pay for the allocation of a unit of each resource in each provider; i's utility is the difference between the total value attributed by i to the allocation and the payments made by i. On the other hand, the valuation v_j of a provider j specifies how much j wants to be paid for allocating a unit of each resource; j's utility is the difference between the payments received by j and the total value attributed by j to the allocation.

Note that in the context of our model, *providers* are the owners of the gateways, and have direct access to the Internet, while *bidders* have no direct access to the Internet and rely on these gateways. The role of

auctioneer can be taken by one of the providers, or a chosen third party, or in the case of our proposed distributed auctioneer, simulated by some providers in a distributed fashion. We consider *resource* to be the bandwidth external to the network available at the gateways.

We will analyse two types of auctions: *standard* and *double* that differ only in who are the bidders (entities that submit bids). In a standard auction, only the users are bidders. Each user i submits a bid b_i to the auctioneer declaring v_i. Then, the auctioneer executes an algorithm $\mathcal{A}$ that returns a feasible allocation and the respective payments. The algorithm $\mathcal{A}$ must satisfy three properties: (1) it must maximise, in expectation, the *social welfare*, defined as the total value attributed by users to the allocation; (2) it must achieve *truthfulness in expectation*, i.e. no user may increase its expected utility by lying in its bid; and (3) it must be *computationally efficient*. In a double auction, the auctioneer collects bids from both the users and the providers. The social welfare is now the difference between the total value of the users and the total value of the providers. In addition to the above three properties, $\mathcal{A}$ should also satisfy *budget balance*, which is the property that the total value paid by users covers the total payments made to the providers. Unfortunately, it was shown that no algorithm can simultaneously satisfy truthfulness in expectation, maximal social welfare, and budget balance [MS83]. In practice, it is common to aim at a combination between truthfulness in expectation and either one of the other two properties.

5.2.2 Distributed Auctioneer Simulation

In our setting, no single entity can be trusted with the role of the auctioneer, since every entity may increase its utility by manipulating the execution of $\mathcal{A}$. Specifically, a provider may devise a sub-optimal schedule that provides him with a higher payment; similarly, a bidder may manipulate the execution of $\mathcal{A}$ to decrease his payment. We address this problem by simulating the role of the auctioneer through a distributed protocol. The idea is to replicate the execution of $\mathcal{A}$ in multiple entities and use cross-validation of the results of the redundant computations. Providers are especially suited for this purpose, since they may be willing to offer their resources for the execution of $\mathcal{A}$ in exchange for payments from the users. Therefore, we focus on distributed protocols executed among sets of providers that deviate from the protocol only if they gain by doing so.

Now, we specify requirements for a correct simulation of the auctioneer. Normally, the auctioneer collects a vector $\vec{b}$ of bids and executes $\mathcal{A}$ with input $\vec{b}$. In a simulation, each provider j must collect a vector $\vec{b}^j$ of bids sent to j and use it as input of a distributed protocol that simulates $\mathcal{A}$. This requires

bidders to submit a bid to all providers. We consider that bidders may adopt arbitrary behaviours such as submitting different bids to different providers or not submitting a bid. Nevertheless, we assume that every provider j eventually collects a vector $\vec{b}^j$ to be used as input in the simulation, containing a bid for every bidder i, and if i is correct, then j receives the bid of i prior to the simulation. In practice, bidders are expected to submit their bids by some deadline; if a bidder fails to do so or sends an invalid bid, then the provider may use the special value $\perp$ instead. We want a simulation of the auctioneer to simulate $\mathcal{A}$ on some input $\vec{b}$ that contains at least the bids sent by correct bidders, regardless of the bids of remaining bidders.

More precisely, let $\mathcal{A}(x, \vec{p} \mid \vec{b})$ be the probability of algorithm $\mathcal{A}$ outputting an allocation x and vector of payments $\vec{p}$, when executed on input $\vec{b}$ by a trusted auctioneer. We denote by b_i^j the bid submitted by bidder i to provider j in a simulation. Let $\vec{b}^j$ be the vector of all bids sent to j. If i does not submit a valid bid to j, then we take b_i^j to be a neutral bid (i.e., a bid that excludes i from the auction). In a simulation of the auctioneer, each provider j inputs $\vec{b}^j$ and outputs a pair $(x, \vec{p})$ composed by an allocation x and a vector of payments $\vec{p}$, or outputs a special value $\perp$ that signals the abortion of the simulation. We say that the outcome is $(x, \vec{p})$ if all providers output this pair, otherwise, the outcome is $\perp$. This has the consequence that the mechanism is not resistant to faults or Byzantine behaviour, since even a single missing or different value in the output can abort the whole allocation process, and we leave it for the the future work (§ 6.2). We assume that an external mechanism guarantees that (1) when the outcome is $\perp$, the auction is aborted, and (2) when the outcome is $(x, \vec{p})$, the allocation x is enforced and all entities perform or receive their respective payments. We can now provide a precise definition of correct simulation.

Definition 5.1. *A simulation is said to be correct if and only if, for all vectors $(\vec{b}^j)_j$, the outcome is $(x, \vec{p})$ with probability $\mathcal{A}(x, \vec{p} \mid \vec{b})$, where $\vec{b}$ only contains valid bids and, for all bidders i such that $b_i^j = b'_i$ for every provider j, we have $b_i = b'_i$.*

5.2.3 Game Theoretical Model

We consider the model of extensive form games played in asynchronous systems proposed in [ADH13]. There are $m > 1$ players corresponding to the providers of the resource allocation auction. Providers may form coalitions of size at most k. We assume that each provider has a unique identifier, known to every other provider. Time is divided into turns. In each turn, some provider j is chosen to move: j

first receives messages sent to it in the previous round, performs some computation, and sends messages. A schedule specifies which provider moves at each turn and which messages it receives. We assume that communication channels are reliable, so every message sent is eventually delivered. We focus on schedules that are *fair* in the sense that every provider j is scheduled to move infinitely often, such that, for all turns t, there exists a turn $t' > t$ when j is scheduled to move. This is necessary to ensure progress.

Now, we want to define a notion of equilibrium for this setting. For this, we need an exact definition of protocol and utility. A protocol specifies, for each schedule, a probability distribution over the computation performed by each provider j and the messages sent at each turn where j moves, as a function of the history of messages sent and received in previous turns. In addition, since we analyse protocols as modules with input and output values, the protocols also specify the values used as input and output by each provider. The utility of providers is a function of the outcome of the simulation: if the outcome is $\perp$, then the utility is 0, else the utility is the difference between the payments received and the value of the allocation. Given this, the utility of a user i is also 0 if the outcome is $\perp$, or is the difference between the value of the allocation and the payments made. The expected utility conditioned on the schedule is computed according to the probability distribution over outcomes induced by the protocol.

For the definition of equilibrium, we consider the notion of k-resilient (ex post) equilibrium introduced in [ADH13]. This notion is a refinement of Nash equilibrium that incorporates collusion and asynchrony. Namely, collusion is modelled as sets K of at most k providers that coordinate on any joint protocol; we require that no provider in K can increase its expected utility if providers in K deviate from the specified protocol, given that other providers do not deviate. Asynchrony is modelled in an ex post way by assuming that providers are informed about the schedule when computing the expected utility. This is the strongest requirement for this setting.

Definition 5.2. *A protocol P is a k-resilient (ex post) equilibrium if and only if for all fair schedules and coalitions K such that $|K| \leq k$, there is no provider in K that increases its expected utility when providers in K follow a joint protocol $P' \neq P$, given that providers not in K follow P.*

An important aspect of this definition is that a k-resilient equilibrium protocol P satisfies the property that no provider increases its expected utility by lying about its input, hence P achieves *truthfulness* regarding the inputs of the providers to a simulation. Specifically, at every invocation, we say that provider j has input v if, by following P, j is expected to input v, so truthfulness implies that j does not input $v' \neq v$. By the truthfulness of $\mathcal{A}$, every bidder i maximises its expected utility by sending $b_i = v_i$ to all

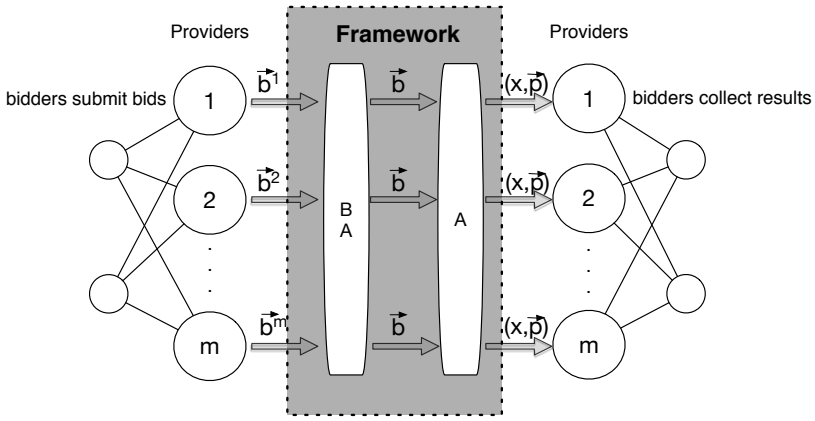


Figure 5.8: Framework: Bid Agreement (BA) and Allocator (A)

providers, regardless of other bids. This implies that to fulfil our goals it suffices to devise protocols that are k-resilient and correctly simulate the auctioneer.

5.3 The Distributed Auctioneer

We propose a framework for devising distributed protocols executed by the providers that correctly simulate the auctioneer. The framework is sufficiently general to simulate different auctions. To illustrate its applicability, we provide two implementations of the framework for standard and double bandwidth allocation auctions, respectively. We describe the framework in two steps. First, we provide a general definition where we do not specify the details about how to implement the simulation of the algorithm $\mathcal{A}$. Then, we describe how to simulate $\mathcal{A}$ by leveraging parallelism to speed up its execution.

5.3.1 General Framework

The input of the framework at each provider j is a vector $\vec{b}^j$ of bids submitted to j and the output is either a pair $(x, \vec{p})$ containing an allocation x and a vector of payments $\vec{p}$ or the special value $\perp$. As illustrated in Figure 5.8, the framework chains the execution of two building blocks: *bid agreement* and *allocator*. Each provider j inputs $\vec{b}^j$ to the bid agreement, which outputs either a vector $\vec{b}$ or $\perp$. In the former case, j inputs $\vec{b}$ to the allocator. If all providers follow the protocol, then the bid agreement ensures that they all output some vector $\vec{b}$ containing all valid bids, and the allocator ensures that they all output a pair $(x, \vec{p})$ with probability determined by $\mathcal{A}$.

In the following paragraphs, we describe each block in more detail by defining properties that must be satisfied by any implementation of the block, and then show in the analysis that every implementation of the framework is k-resilient and correctly simulates the auctioneer based only on the properties of the blocks. This makes the proof independent from the actual implementation. In all blocks, an implementation P must satisfy the property of *k-resiliency for solution preference*, i.e., P must be a k-resilient equilibrium, under the assumption that players have preference for a solution and number of agents not in the same coalition is sufficiently high. Specifically, the output of every block is either some valid value or $\perp$. We can split the set of outcomes of the block (combinations of outputs) into the set A of solutions where all providers output the same valid value and the set B of remaining outcomes. In a correct execution, we want the outcome to lie in A. To ensure this and that the protocol is a k-resilient equilibrium, we need to assume that providers obtain a higher utility for outcomes in A than for outcomes in B (preference for a solution), and $m > f(k)$ for some function f defined for every $k > 0$. The assumption of preference for a solution of the framework is equivalent to providers preferring to receive the payments.

Bid Agreement

The input at provider j is the vector $\vec{b}^j$ of bids sent to j. The output is a vector $\vec{b}$ or the special value $\perp$. In addition to k-resiliency for solution preference, this block must ensure two conditions when all providers follow the protocol: (1) *eventual agreement*, defined as all providers eventually outputting the same vector $\vec{b}$, and (2) *validity*, defined as, for every bidder i that submits the same bid b'_i to all providers, the output at every provider is $b_i = b'_i$.

Property 5.1. *A protocol P implements bid agreement if and only if it satisfies two conditions: (1) if all providers follow P, then P satisfies eventual agreement and validity; and (2) k-resiliency for solution preference.*

If we can assume that the bids of malicious bidders are obtained from a finite set of values and are equally likely, then a suitable approach is to use the rational consensus protocol proposed in [Afe+14], which has inputs $\{0, 1\}$ and outputs in $\{0, 1, \perp\}$, and satisfies the following two properties: (a) if all providers follow the protocol, then all providers eventually output the same bit, which is input by some provider; and (b) k-resiliency for solution preference, assuming $m > 2k$ and that the input of every provider not in the same coalition is either the same value or is 0 or 1 with equal probability. This protocol can be used to implement the bid agreement as follows. For each bidder i, provider j generates a stream

of bits uniquely determined from b_i^j and inputs each bit to a rational consensus instance; if some instance outputs $\perp$, then j outputs $\perp$, otherwise, j converts the stream to a bid b_i and outputs a bid b_i^*, where $b_i^* = b_i$ if b_i is valid, or b_i^* is some pre-determined valid bid otherwise. To distinguish between different instances of rational consensus, providers may append to the messages of each instance the identifier of each bidder and the position of each bit. Clearly, providers only output valid bids or the value $\perp$. By (a), if all providers follow the protocol, then eventual agreement and validity hold, showing (1). Condition (2) follows directly from (b) and $m > 2k$ if the input of every provider satisfies the assumptions of (b). To see why these assumptions are true, notice that, for each bidder i, if i is not malicious, the input of all providers not in the same coalition is i's true bid, and if i is malicious, then the bid b_i^j sent by i to j is uniformly distributed. If the set of possible bids is the set of all integers, then the stream of bits obtained from b_i^j is also random. These are reasonable assumptions, since we expect the behavior of malicious bidders to be arbitrary.

Allocator

The input at every provider is a vector $\vec{b}$ of bids, and the output is either a pair $(x, \vec{p})$ or $\perp$. We want the allocator to satisfy four conditions. First, we want the allocator to correctly simulate $\mathcal{A}$, i.e., given that all providers input the same vector $\vec{b}$ and follow the protocol, every provider must eventually output pair $(x, \vec{p})$ with probability $\mathcal{A}(x, \vec{p} \mid \vec{b})$. Second, we want resilience to collusive influences, defined as, for all coalitions K of at most k elements, if all providers not in K input $\vec{b}$ and follow the protocol, then no $j \notin K$ outputs a pair $(x, \vec{p})$ with probability higher than $\mathcal{A}(x, \vec{p} \mid \vec{b})$, regardless of the protocol followed by providers in K. Intuitively, no coalition K can influence the output of providers not in K, except that they may output $\perp$ with higher probability. Third, we want input validation to ensure that providers have preference for solutions at the bid agreement. More precisely, if two providers input different vectors and follow the protocol, then they both output $\perp$, regardless of the protocol followed by other providers. Finally, we want k-resilience for solution preference given that all providers have the same input.

Property 5.2. *A protocol P implements the allocator if and only if it satisfies four conditions: (1) correct simulation of $\mathcal{A}$; (2) resilience to collusive influence; (3) input validation; and (4) k-resiliency for solution preference if all providers have the same input.*

We discuss implementations of the allocator in § 5.3.2.

Analysis

We show in Theorem 5.1 that a protocol that implements our framework correctly simulates the auctioneer and is k-resilient.

Theorem 5.1. *For every protocol P that implements the framework, P correctly simulates the auctioneer, and there exists a function f such that, if $m > f(k)$, then P is a k-resilient equilibrium.*

Proof. First, we show that P correctly simulates the auctioneer. Every provider j inputs $\vec{b}^j$ to the bid agreement. By (1) of Property 5.1, regardless of the inputs, all providers output the same vector $\vec{b}$ that satisfies validity. By (1) of Property 5.2, the outcome of the simulation is pair $(x, \vec{p})$ with probability $\mathcal{A}(x, \vec{p} \mid \vec{b})$. This concludes the first step of the proof.

Now, we show that P is a k-resilient equilibrium for $m > f(k)$ for some f. Fix a coalition K. We take f to be larger for all k than the minimum value of m required by Properties 5.1 and 5.2. These properties imply that, if providers have preference for a solution at the bid agreement, then the implementations of bid agreement is k-resilient, so providers in K prefer to follow P for bid agreement. Since this guarantees that all providers have the same input at the allocator, the implementation of the allocator is also k-resilient, implying that P is k-resilient.

Now, we show that solution preference holds for both blocks. Recall that the outcome is not $\perp$ only if providers not in K output the same pair, and, if the outcome is $\perp$, then the utility is 0. Hence, providers in K prefer (obtain an expected utility at least as high) that providers not in K output the same pair $(x, \vec{p})$; in this case, they clearly prefer to output $(x, \vec{p})$ as well, thus they have preference for a solution at the allocator. Now, consider the bid agreement. The utility of an outcome of this block is the expected utility given that providers not in K follow P and providers in K follow an arbitrary protocol. Clearly, providers in K prefer that no provider in K outputs $\perp$. By (3) of Property 5.2, providers in K prefer that all providers not in K output the same vector. By (2) of Property 5.2, providers in K cannot increase the probability of any outcome of the framework other than $\perp$ by deviating, thus, they cannot increase their expected utility by outputting a vector $\vec{b}' \neq \vec{b}$ at the bid agreement. This shows that providers have preference for a solution at the bid agreement, concluding the proof. □

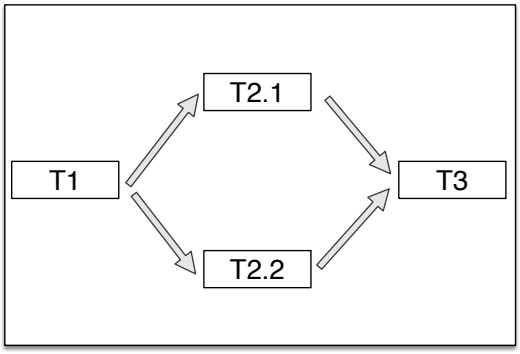


Figure 5.9: Decomposition of the Allocator into Tasks

5.3.2 Parallel Allocator Framework

We describe a framework for implementations of the allocator that satisfy Property 5.2. We explore the possibility of parallelising the execution of $\mathcal{A}$ in multiple providers. Although this approach introduces the overhead of communication between providers, since $\mathcal{A}$ is often computationally intensive, its parallelisation compensates for this overhead.

The framework consists in an initial invocation of a building block for *input validation* followed by the simulation of $\mathcal{A}$, which invokes two additional building blocks: *data transfer* and *common coin*. The input is a vector of bids and the output is either $\perp$ or a pair $(x, \vec{p})$. At the invocation of each block, providers either output a valid value or $\perp$; in the latter case, they output $\perp$ at the allocator. To describe the simulation of $\mathcal{A}$, it is useful to characterise the execution of $\mathcal{A}$ in terms of a graph of tasks, where nodes correspond to tasks to be executed in sequence and edges represent data dependencies. This graph establishes a partial order of tasks; every two tasks that are not ordered can be executed in parallel by different providers. Figure 5.9 gives an example of a graph of 4 tasks, where tasks T2.1 and T2.2 can be executed in parallel. To cope with collusion, each task T is assigned to a set S of at least $k + 1$ providers. If a task T' is to be executed by a set $O \neq S$ of providers and T' depends on the result of T, then the providers of S transfer data to the providers of O using the data transfer building block. In a correct simulation of $\mathcal{A}$, there must be one final task that depends on all other tasks, where all providers gather all the required data to produce the final output. Whenever providers need a random number distributed according to a probability distribution Π, they invoke the common coin with input Π. Figure 5.10 illustrates the framework for the task decomposition of Figure 5.9.

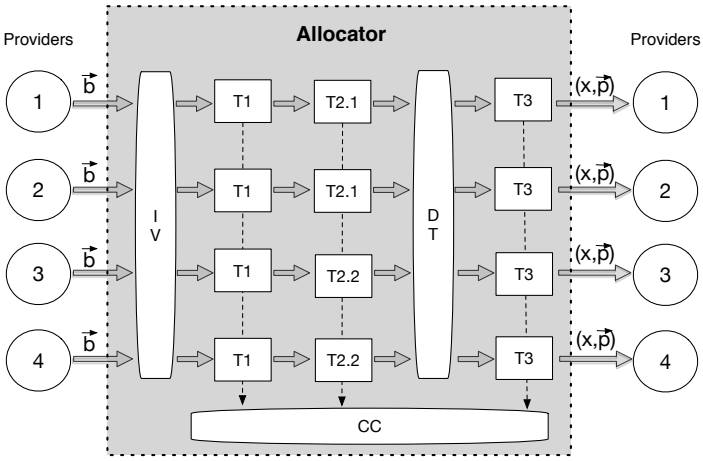


Figure 5.10: Parallel Allocator: Input Validation (IV), Data Transfer (DT), and Common Coin (CC)

As in the previous section, we describe properties that must be satisfied by the implementations of each block and then show that every implementation of this framework satisfies Property 5.2.

Input Validation

The input is a vector $\vec{b}$ and the output is either $\perp$ or $\vec{b}$. We want an implementation to satisfy k-resiliency for solution preference and that all providers eventually output $\vec{b}$ given that they all input $\vec{b}$, and we need to satisfy (3) from Property 5.2.

Property 5.3. *An implementation P of the input validation must satisfy three conditions: (1) if two providers follow P and have different inputs, then they eventually output $\perp$; (2) if all providers follow P with the same input $\vec{b}$, then they eventually output $\vec{b}$; and (3) k-resiliency for solution preference if all providers have the same input.*

A simple implementation is to have providers broadcasting their vectors of bids and outputting $\perp$ when two different vectors are detected. This clearly satisfies (1) and (2), whereas (3) is immediately true if providers have preference for a solution and $m > k$.

Common Coin

The input is a probability distribution Π and the output is either $\perp$ or a number distributed according to Π. Given that all providers have the same input, we want the common coin to satisfy k-resilience for

solution preference and to output the same random number.

Property 5.4. *Given that all providers have input Π, an implementation P of the common coin must satisfy two conditions: (1) if all providers follow P, then they eventually output the same value distributed according to Π; and (2) k-resiliency for solution preference.*

A possible implementation of the shared coin is the protocol from [ADH13]. The idea is that every provider j commits to a random number $r_j \in [0, 1]$, before learning the random numbers of every other provider not in its coalition. Then, providers reveal all random numbers and compute the output by summing all numbers modulo 1. If some provider j sees a number not in $[0, 1]$ or some provider does not send a random value compatible with its commitment, then it outputs $\perp$. Otherwise, j applies a transformation on the computed value, which is uniformly distributed in $[0, 1]$, to produce an output that is distributed according to Π.

It is clear that all providers output the same random number distributed according to the common input Π if they follow the protocol. Assuming that $m > k$, no provider j can manipulate the probability distribution of the output by not committing to r_j selected at random without some provider outputting $\perp$, even if j is in a coalition of at most k providers. Therefore, the protocol satisfies k-resiliency for solution preference.

Data Transfer

A set S of providers inputs a value from a domain D. Providers from a set O either output a value from D or $\perp$. When all providers in S have the same input, we want them to output the same value in D when they follow the protocol. We only require an implementation to be k-resilient if $|S|, |O| > k$, since otherwise coalitions can always manipulate the output of this block.

Property 5.5. *Given that $|S|, |O| > k$ and all providers have the same input v, an implementation P of the data transfer must satisfy two conditions: (1) if all providers follow P, then they eventually output v; and (2) k-resiliency for solution preference.*

We propose a simple k-resilient implementation of this block, where providers in S broadcast their input to all providers in O. In the end, if some provider $j \in O$ detects two different values, then j outputs $\perp$. Given that all providers have input v and that $|S|, |O| > k$, they eventually output v, and no coalition

K of up to k providers can cause all providers to output $v' \notin \{v, \perp\}$. By solution preference, no provider in K gains if someone lies about the input v or omits a message.

Analysis

Theorem 5.2 shows that every implementation of the above framework satisfies the four conditions of Property 5.2.

Theorem 5.2. *Every protocol P that implements the parallel allocator satisfies Property 5.2.*

Proof. We show that P ensures (1) correct simulation of $\mathcal{A}$; (2) resilience to collusive influence; (3) input validation; and (4) k-resiliency for solution preference if all providers have the same input. First, we show (1). Suppose that all providers input the same vector $\vec{b}$ and follow P. We show that every provider outputs the same pair $(x, \vec{p})$ with probability $\mathcal{A}(x, \vec{p} \mid \vec{b})$. We show using induction that, if the decomposition of $\mathcal{A}$ into tasks is done correctly and we fix all random numbers, then at every task T every provider j that executes T has the same output that she would have if j executed $\mathcal{A}$ locally with the same random numbers. This is true for the first task by (2) of Property 5.3. In the inductive step, the input at each task depends only on the output of a set of tasks. For each of those tasks T, by the induction hypothesis, a set S of at least $k + 1$ providers computes the same result and inputs it to the data transfer; by (1) of Property 5.5, all providers that execute T receive that value and perform the same computation as they would if they were executing $\mathcal{A}$. This implies that all providers output the same pair at the end. By (1) of Property 5.4, at every invocation of the common coin, all providers input the same distribution Π and output the same random number distributed according to Π, where Π is specified by $\mathcal{A}$. This proves (1).

Now, we show (2). Fix a coalition K and suppose that all providers not in K follow P with input $\vec{b}$ and providers in K follow an arbitrary $P' \neq P$. The only way that providers in K could cause providers not in K to return pair $(x, \vec{p})$ with probability higher than $\mathcal{A}(x, \vec{p} \mid \vec{b})$ is if the result of some task used in the input of another task or as the final output is not distributed as specified by $\mathcal{A}$ and $\vec{b}$. Since each task is executed by more than k providers, using an identical reasoning to the proof of (1), we can show using induction that providers in K cannot manipulate the probability distribution over the results of each task, except only by increasing the probability of some provider not in K outputting $\perp$. Here, we use the fact that, by (3) of Property 5.3 and (2) of Properties 5.4, 5.5, providers in K cannot manipulate the probability distribution over outputs of the building blocks in a way that increases the expected utility of some provider in K. This proves (2).

Condition (3) follows by (2) of Property 5.3. To show (4), we first need to prove that providers have preference for a solution at all invocations of building blocks, assuming that they have preference for a solution of the allocator. Fix a coalition K. It is clear that providers in K prefer that providers not in K do not output $\perp$ at all invocations. Now, we use backward induction to show that they prefer that providers not in K never return different values. In the last invocation, this is clearly true by preference for a solution of the allocator. Continuing backwards, if two providers not in K output different values at the same invocation of some block, then either they output different pairs at the end or input different values at the following invocation of the data transfer, which by the hypothesis is never preferable to outputting the same value at the considered invocation. By the proof of (2), providers in K cannot manipulate the final outcome by not outputting the same values at all invocations, so they also prefer to output the same values as providers not in K, showing solution preference at all invocations. This also shows that providers prefer to have the same input at all invocations. Thus, given that all providers have the same input, no provider in K can increase its expected utility if some provider $j \in K$ does not compute each task correctly. By (3) of Property 5.3 and (2) of Properties 5.4 and 5.5, P is a k-resilient equilibrium. □

5.3.3 Resource Allocation Instances

We now show how our framework can be applied to two different bandwidth allocation problems in the context of community networks. For that purpose, we resort to two different algorithms that have been proposed in the literature to solve bandwidth allocation for users in providers. These algorithms rely on standard and double auctions respectively, and have different computational properties: the double auction algorithm provides an example of a graph with only one task that is not computationally intensive, such that decomposing its execution into parallel tasks does not provide a performance gain; the standard auction algorithm provides a graph with multiple computationally intensive tasks that can be parallelised. Later in the chapter (§ 5.4), we will use these examples to evaluate the performance of implementations of the framework. We use the double auctions example to measure a worst-case overhead of executing all building blocks of the framework compared to an execution with a centralised trusted auctioneer, and we use the standard auctions example to show that the improvements of parallelisation can outweigh the added overhead when the execution time is dominated by computation.

Double Auction

Consider an auction where each provider has a limited bandwidth to be allocated to multiple users, and each user has a demand of bandwidth that may be satisfied by multiple providers. Both the users and the providers declare in their bids the value given to a unit of allocated bandwidth. An allocation gives the amount of bandwidth for each user allocated in each provider. We want to ensure truthfulness in expectation and budget balance. For this purpose, we use the algorithm $\mathcal{A}$ of [Zhe+14], which provides the above properties at the expense of social welfare. The idea is to order the providers by increasing value and to order the users by decreasing value. Then, users are allocated by their order to the providers using the water-filling method: the maximum amount of bandwidth of each user is allocated to the first available provider without exceeding its capacity, and any unsatisfied demand of that user is allocated to the following providers using the same method. Since the most computationally intensive task of this algorithm is sorting, in most practical settings there is no performance gain in parallelising the execution of $\mathcal{A}$. Instead, every provider executes $\mathcal{A}$ locally and outputs the result. Hence, we never need to invoke the data transfer building block.

Standard Auction

Consider a variation of the double auction where providers do not send bids and each bidder can only have its bandwidth demand allocated in a single provider. Here, we aim for truthfulness in expectation, maximal social welfare, and computational efficiency. It is well known that a VCG mechanism can be used to provide the first two guarantees. The difficulty is that determining the maximal social welfare is in general an NP Hard problem, which conflicts with the goal of computational efficiency. To address this issue, we use the algorithm of [Zha+15b] which adapts the VCG mechanism to achieve a tradeoff between the two conflicting requirements. Specifically, [Zha+15b] offers a $(1-\epsilon)$ approximation of maximal social welfare for an arbitrarily small ϵ, while terminating in polynomial time according to smoothed analysis.

Interestingly, the randomised algorithm proposed in [Zha+15b] has the potential for parallelisation. In a course manner, the algorithm can be divided into three steps, depicted in Algorithm 5.2. The first step derives an approximately optimal allocation of users to providers. This step is hard to parallelise effectively in a distributed system, so we run it in a single sequential task. The second step calculates the payments for each user based on the result of the first step. This step is computationally intensive and the payments for each user can be computed independently and, therefore, can be easily parallelised. The final step

Algorithm 5.2 Standard auction allocator

1: Task 1: Calculate the allocation solution x
2: **for** Each subset S of bidders in parallel **do**
3: Task 2.S: calculate payment p_j of every $j \in S$
4: **end for**
5: Task 3: Collect the outputs of each task with the data transfer and output $(x, \vec{p})$

gathers all intermediate results to produce the output. In our implementation, the first and third steps are executed by all providers. In the second step, we group the providers into c groups, each containing at least $k + 1$ providers. Each group is assigned the computation of the payments of a subset of n/c users. Then, all providers of a group execute the data transfer block to transfer the resulting payments to all providers.

5.4 Performance Evaluation

We have evaluated the implementations of the allocator for double and standard auctions proposed in § 5.3.3. The implementations of all the remaining blocks are as suggested in § 5.3.2: we use the rational consensus algorithm proposed in [Afe+14] in the implementation of the bid agreement, while the input validation and data transfer blocks are implemented as simple broadcasts, and the common coin is implemented using the scheme from [ADH13]. For these implementations to be k-resilient equilibria, we need $m > 2k$. This is a requirement of the rational consensus algorithm.

Our goal is to assess the overhead of the distributed protocol, when compared to a purely centralised solution, in the case the allocation algorithm is not parallelisable, and to assess the potential benefits from parallelisation when computationally expensive allocation algorithms are used. For that purpose, we measure performance gains for different levels p of parallelism, where $p = \lfloor m/(k+1) \rfloor$ is the maximum level of parallelism for each possible k and $p = 1$ represents the sequential execution by a trusted auctioneer. We consider a fixed number of $m = 8$ providers in the auctions, but we vary the number of providers that execute each protocol.

5.4.1 Hardware/Software Setup

In order to obtain a meaningful evaluation of our approach, we have resorted to a prototype implementation on realistic hardware and software environment and deployed it in an experimental testbed for

community networks, namely on nodes of the Guifi.net, one of the largest community networks in the world [Com16a]. We were given access to 4 different nodes of the experimental testbed, 2 machines in Barcelona (UPC Campus), and one machine each in Barcelona (Hangar) and in Taradell, Spain. When doing tests executed by more than 4 providers we have instantiated multiple VMs in each nodes, ensuring that each VM is allocated a different CPU. The machines are Intel Core i7-3770 3.40GHz CPU with 16 GB RAM and 1 TB hard disk, running Proxmox virtualisation engine. Our experiment runs in OpenVZ containers with Debian 7 x86 1 CPU, 2 GB RAM, and 10 GB storage. We have implemented the framework in Python, using PyPy for speed reasons, and used ØMQ [Zmq16] as the messaging library for the communication.

We have set up a single node that acts as a client, and generates input for all the n users. This client node sends the requests to the m providers, and receives the results back from all of them. The values for running time presented in the plots capture the time from when the inputs are generated at this client node, till the time it receives the results from all the experiment instances. We run the experiments for 100 rounds, and plot the average values in the graphs.

5.4.2 Double Auction Deployment

We have used an experimental set up similar to [Zhe+14], with some slight modifications suitable to our use case. In both the experiments, the double and standard auction, the bids by the users are uniformly distributed in the range $[0.75, 1.25]$, and the requested bandwidth resource is uniformly distributed in the range $(0, 1]$. We vary the capacity of the providers depending upon the overall bandwidth required, and scale it using a random factor in $[0.5, 1.5]$ so as to consider both the cases where providers lack the capacity to satisfy all the requests, and where the providers have excess capacity. The providers have a unit cost of bandwidth uniformly distributed in the range $(0, 1]$.

Figure 5.11 shows the running time for the double auction algorithm (§ 5.3.3) as a function of the number of users, for up to 1000 users. This algorithm has little computational overhead. It is not easily parallelisable but, as it can be observed from the figure, this is irrelevant as the distributed version is dominated by the communication time. Also, the communication overhead increases as the number of users increases, since more data has to be exchanged between the providers. The figure shows the values obtained for the centralised approach and for the distributed implementation using different values of k and corresponding minimum required number of providers out of a total of 8 involved in the execution,

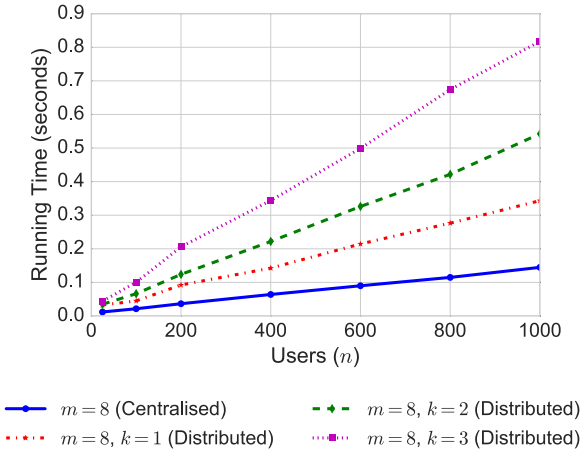


Figure 5.11: Running time for double auction

namely, 3 providers when $k = 1$, 5 when $k = 2$, and 8 when $k = 3$. Even when 8 providers and 1000 users are used, the distributed implementation finishes in less than a second which is perfectly acceptable because normally these auctions need to run with reasonable intervals between them.

5.4.3 Standard Auction Deployment

We fix the number of providers to $m = 8$ and vary the maximum degree of parallelisation by taking p to be 1, 2, and 4, corresponding to a centralised execution, $k = 3$, and $k = 1$, respectively. Figure 5.12 shows the running time for the standard auction (§ 5.3.3) as a function of the number of users, for up to 125 users. The capacity of the providers is based on the overall bandwidth required at that provider in the bids submitted by the users, and scaled down using a random factor in $[0, 0.25]$, so roughly no more than a quarter of the users win the bids. For higher values of n, the algorithm [Zha+15b] can take in the order of hours to complete, which is expected as its computational complexity is $\approx \mathcal{O}(mn^9(\frac{1}{\epsilon})^2)$ for n users and m providers, though it provides better guarantees for social welfare than other alternatives.

Figure 5.12 shows that the running time in general grows quickly as n increases, and there is sharp rise in the running time for values of n close to 100. This is because the running time of the algorithm [Zha+15b] is a function of the feasible allocation space (which can grow exponentially in the worst case) of the resource allocation problem. Therefore, the communication and coordination overhead is not significant when compared to the running time of the allocation algorithm. On the contrary, the overheads involved in distributing the inputs and aggregating the results from the providers are easily off-

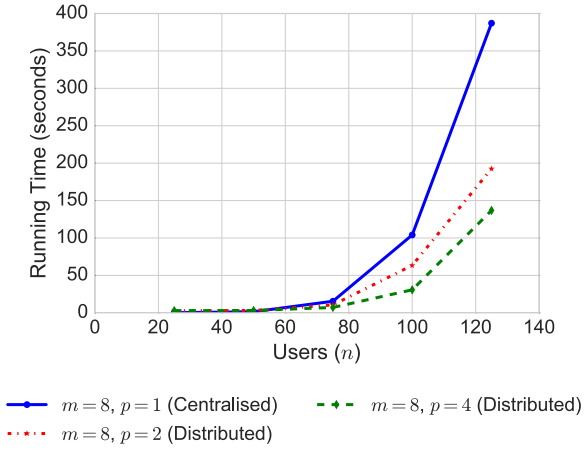


Figure 5.12: Running time for standard auction

set by the gains due to parallelisation in this case. In Figure 5.12, we can observe significant performance gains in the distributed case, for $p = 2$ and $p = 4$ (i.e., for $k = 3$ and $k = 1$ respectively). For instance, when 8 providers are available and $k = 1$ the distributed implementation takes around 100 seconds while the serial implementation takes around 400 seconds. This indicates that our approach allows for scaling the allocation algorithm, given that when the network grows more providers also become available.

5.5 Summary

Resource allocation is a fundamental problem in networked systems and the design of auction mechanisms that can provide properties such as truthfulness, budget balance, and maximal social welfare have been extensively studied in the literature. These works assume a centralised trusted auctioneer that can faithfully execute the allocation algorithm. Unfortunately, many networked systems of today, such as "clouds of clouds", edge clouds, and community networks, among others, lack a central trusted point of control (and, if it existed, it would be a bottleneck). In this chapter, we have addressed the theoretical and practical challenges that need to be overcome to bridge this gap. More precisely, we have proposed a novel distributed framework for devising Nash equilibria distributed simulations of the auctioneer that are resilient to asynchrony and coalitions. Furthermore, our framework allows for the parallelisation of the allocation algorithm, leveraging the distributed nature of the simulation, which is of paramount practical importance given that, in many allocation algorithms, achieving maximal social welfare is computationally intensive. We have devised implementations of the framework in a realistic testbed of one of the

largest community networks deployed today, and have gathered experimental evidence that the overhead of the emulation is not significant, even in the cases the allocation algorithm cannot be parallelised, and brings substantial gains in the case parallelisation is possible. This shows that our approach can be used as a building block to implement resource allocation in decentralised networks.

Notes

The results presented in this chapter were accomplished in cooperation with my co-advisor Luís Rodrigues and Xavier Vilaça, another PhD student at IST. Xavier did provide relevant contributions to the design of the game theoretical framework.

The study on the need for incentive-compatible pricing mechanisms (§ 5.1) was presented in the paper "*Towards Incentive-Compatible Pricing for Bandwidth Reservation in Community Network Clouds*" [Kha+15b], in 12th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2015), Cluj-Napoca, Romania, September 2015. The results for Distributed Auctioneer were published as a full paper "*A Distributed Auctioneer for Resource Allocation in Decentralized Systems*" [Kha+16b], in 36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016), Nara, Japan, June 2016.

6

Conclusion

Cloud computing with its success in providing virtualised resources on demand has transformed the technology landscape, revolutionising how Internet applications are developed and delivered to the users. Peer-to-peer and edge computing models have been explored in the past decade, but other than a few success stories they never made it big in the mainstream. Perhaps now is the right opportunity to take full advantage of the virtualization model of the cloud computing to design the killer applications for the community cloud. On the technical side, this allows for sophisticated applications and services that were not possible with the simple process-level isolation approaches of earlier efforts like BOINC or Seattle. The challenge is to provide developer tools and middleware services that streamline the process of programming and deploying the community cloud applications. At the same time, killer applications are needed that by satisfying users' critical needs and problem scenarios succeed in engaging the community for the long-term.

In this thesis, we looked at the field of community clouds in general, and clouds in community networks in particular, to develop economic regulation mechanisms in tune with the specific social, economic and technical context of the community networks in order to help us in developing and sustaining a com-

munity cloud ecosystem. We looked at how these mechanisms fit in an overall framework of a community cloud system, which supports components for incentivising contribution, regulating access to resources, and ensuring trust in the system. We developed incentive-based resource regulations mechanisms for a well-knit community of trusted users, and showed how they are crucial for successful operation of a community cloud. We also noticed how lack of trust in a community cloud deployed on a large scale affects negatively the utility and sustainability of the system. To address this issue, we developed a distributed auctioneer component, which is a virtual trusted entity integrated in the overall architecture of the community cloud. This component allows to efficiently and optimally allocate resources in a community of untrusted users, with negligible communication and computation overhead, and it can also leverage parallelised implementation to offer scalability.

6.1 Ramifications and Collaborations

The research in this thesis was extended in other contexts through collaborations. We discuss the details below, and comment on their relevance to the work presented in this thesis.

6.1.1 Community Clouds

We study the general idea of community clouds in the context of other prevailing cloud models, in particular for the industry [KFN16]. We investigate how this extends to community clouds built collaboratively, realising ideas from edge computing and volunteer computing [KFR15]. This provides the broad context for community network clouds, and indicates their potential use cases.

6.1.2 Social and Economic Mechanisms

We explore the social and economic mechanisms that can help in adoption and growth of community network clouds [KF14a; KF14b]. We take into account the social and technical context of community networks, and present a cost-value proposition describing the conditions under which community network clouds could emerge. We propose a set of technical, social and economic mechanisms that, if placed in community networks, can help accelerate the uptake and ensure the sustainability of community network clouds. These mechanisms highlight the importance of managing incentives in community network clouds.

6.1.3 Scalability of Community Cloud Architectures

We investigate the scalability of community network cloud model [Kha+13] in order to analyse different aspects of its performance in comparison to public clouds through simulation experiments. In a community network cloud the computing resources are heterogeneous and less powerful, but are geographically distributed and are located closer to the users. Our results suggest that the performance of the community clouds depends on the conditions of the community networks, but has potential for improvement with network-aware cloud services.

6.1.4 Supporting Service Selection

Community network clouds need support mechanisms that provide assistance in cloud service selection while taking into account different aspects pertaining to associated risks in community clouds, quality concerns of the users and cost limitations specifically in multi-clouds ecosystems. We propose a risk-cost-quality based decision support system [Kha+15a] to assist the community cloud users to select the most appropriate cloud services meeting their needs. The proposed framework not only increases the ease of adoption of community clouds by providing assistance to users in cloud service selection, but also provides insights into the improvement of community clouds based on user behaviour.

6.1.5 Cloud Services in Guifi.net

We materialise the proposed framework for community network cloud [Jim+13; Fre+14; Bai+15a; Sel+15; Kha+16a] in the implementation of the *Cloudy* distribution [Clo16]. This distribution can be used to integrate useful services and applications that provide value to end-users of the community network cloud. We conduct real deployments of these clouds in the Guifi.net community network and evaluate cloud-based applications such as service discovery and distributed storage. This deployment experience supports the feasibility of community clouds, and our measurements demonstrate the performance of services and applications running in these community clouds. Our results encourage the development and operation of collaborative cloud-based services using the resources of a community network, and we anticipate that such services can effectively complement commercial offers.

6.2 Future Work

Carrying onwards from the experience and results with the prototype implementations, a working service needs to be developed further, that provides the feedback loop between the users' contribution and experience, and will be inevitable for adoption, sustainability, maintenance and growth of cloud infrastructures in community networks. Larger scale deployments are required with extended implementation of the different components of the community cloud framework. This should be complemented by additional services and applications deployed in the cloud infrastructure, which will provide enhanced value and utility to the members of community networks for their contribution towards the community cloud.

With respect to bandwidth allocation mechanism, there are others challenges, for instance, multiple users may be connected to the provider using the same path in the community network, and reserving bandwidth for such users in the same time interval may cause congestion across some of the links, which an intelligent allocation algorithm should try to avoid. Moreover, any bandwidth reservation scheme should not negatively impact the normal operation of the community network, so allocation mechanism needs to be adaptive to the network congestion and bandwidth usage in the community network.

For distributed auctioneer, we have considered only rational users, and we plan to extend our framework to the Byzantine users. In the current model, we assumed all providers to be fully inter-connected. But in the case of federated community clouds, it is possible that providers in different local clouds may not have very good connections between them, and so the current approach may result in slow down. In such cases, we plan to explore how to adapt our distributed auctioneer to different network topologies.

The area of community cloud builds on a vast body of research in peer-to-peer and edge computing research, and the various lessons learnt from the successes and the failures of many P2P applications. We see a huge opportunity in extending this work for building the core community cloud services that drive innovation in many related areas, and not just the edge computing. The determinant of this success will not be just the technical sophistication with which the research challenges and open problems are solved, but also by how well the enthusiasts of the community cloud succeed in capturing the imagination and meeting the expectations of the end users.

Notes

The research discussed in this chapter (§ 6.1) was included in the following publications.

[Bai+15] Roger Baig, Felix Freitag, Amin M Khan, Agusti Moll, Leandro Navarro, Roger Pueyo Centelles, and Vladimir Vlassov. "Community Clouds at the Edge deployed in Guifi.net". In: *4th International Conference on Cloud Networking (CloudNet 2015)*. Niagara Falls, Canada: IEEE, Oct. 2015.

[Fre+14] Felix Freitag, Leila Sharifi, Amin M Khan, Leandro Navarro, Roger Baig, Pau Escrich, and Luis Veiga. "A Look at Energy Efficient System Opportunities with Community Network Clouds". In: *Workshop on Energy-Efficient System (EES), within 2nd International Conference on ICT for Sustainability (IST4S 2014)*. Stockholm, Sweden, Aug. 2014.

[Jim+13] Javi Jiménez, Roger Baig, Pau Escrich, Amin M Khan, Felix Freitag, Leandro Navarro, Ermanno Pietrosemoli, Marco Zennaro, Amir H Payberah, and Vladimir Vlassov. "Supporting cloud deployment in the Guifi.net community network". In: *5th Global Information Infrastructure and Networking Symposium (GIIS 2013)*. Trento, Italy: IEEE, Oct. 2013.

[KF14a] Amin M Khan and Felix Freitag. "Exploring the Role of Macroeconomic Mechanisms in Voluntary Resource Provisioning in Community Network Clouds". In: *11th International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2014)*. Vol. 290. Advances in Intelligent Systems and Computing. Salamanca, Spain: Springer International Publishing, June 2014, pp. 269–278.

[KF14b] Amin M Khan and Felix Freitag. "Sparks in the Fog: Social and Economic Mechanisms as Enablers for Community Network Clouds". In: *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* 3.8 (2014).

[Kha+15] Amin M Khan, Felix Freitag, Smrati Gupta, Victor Muntés-Mulero, Jacek Dominiak, and Peter Matthews. "On Supporting Service Selection for Collaborative Multi-Cloud Ecosystems in Community Networks". In: *29th IEEE International Conference on Advanced Information Networking and Applications (AINA 2015)*. Gwangju, Korea, Mar. 2015.

[KFN16] Amin M Khan, Felix Freitag, and Leandro Navarro. "Community Clouds". In: *Encyclopedia of Cloud Computing*. Ed. by San Murugesan and Irena Bojanova. Wiley-IEEE, June 2016.

[Kha+16] Amin M Khan, Felix Freitag, Leandro Navarro, and Roger Baig. "Enabling Clouds in Community Networks". In: *European Project Space on Research and Applications of Information and Communication Systems*. Ed. by Carlos Cerqueira and James Uhomoibhi. Lisbon, Portugal: SCITEPRESS, 2016.

[KFR15] Amin M Khan, Felix Freitag, and Luis Rodrigues. "Current Trends and Future Directions in Community Edge Clouds". In: *4th International Conference on Cloud Networking (CloudNet 2015)*. Niagara Falls, Canada: IEEE, Oct. 2015.

[Kha+13] Amin M Khan, Leila Sharifi, Leandro Navarro, and Luis Veiga. "Clouds of Small Things: Provisioning Infrastructure-as-a-Service from within Community Networks". In: *2nd International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2013), within IEEE WiMob*. Lyon, France: IEEE, Oct. 2013, pp. 16–21.

[Sel+15] Mennan Selimi, Amin M Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. "Cloud services in the Guifi.net community network". In: *Computer Networks* 93.P2 (Dec. 2015), pp. 373–388.

Bibliography

[Abr+06] I Abraham, D Dolev, R Gonen, and J Halpern. "Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation". In: *PODC*. 2006, pp. 53–62 (cit. on pp. 16, 17).

[ADH13] Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. "Distributed Protocols for Leader Election: A Game-Theoretic Perspective". In: *DISC*. Vol. 8205. LNCS. Jerusalem, Israel, Oct. 2013, pp. 61–75 (cit. on pp. 16, 17, 51, 63, 64, 71, 75).

[Afe+14] Yehuda Afek, Yehonatan Ginzberg, Shir Landau Feibish, and Moshe Sulamy. "Distributed computing building blocks for rational agents". In: *PODC*. New York, NY, USA: ACM Press, July 2014, pp. 406–415 (cit. on pp. 16, 17, 66, 75).

[Agm+13] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Assaf Schuster, and Dan Tsafrir. "Deconstructing Amazon EC2 Spot Instance Pricing". In: *ACM Transactions on Economics and Computation* 1.3 (Sept. 2013), pp. 1–20 (cit. on pp. 11, 15).

[Aiy+05] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. "BAR fault tolerance for cooperative services". In: *ACM SIGOPS Operating Systems Review* 39.5 (Oct. 2005), p. 45 (cit. on p. 16).

[Alb04] Michael Albert. *Parecon: Life After Capitalism*. Verso Books, 2004 (cit. on pp. 8, 29, 33).

[And04] David P Anderson. "BOINC : A System for Public-Resource Computing and Storage". In: *5th IEEE/ACM International Workshop on Grid Computing*. Pittsburgh, USA, Nov. 2004, pp. 4–10 (cit. on pp. 7, 21).

[And+02] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. "SETI@home: an experiment in public-resource computing". In: *Communications of the ACM* 45.11 (Nov. 2002), pp. 56–61 (cit. on pp. 7, 21).

[ALS10] J Chris Anderson, Jan Lehnardt, and Noah Slater. *CouchDB: The Definitive Guide*. 1st. O'Reilly Media, Inc., 2010 (cit. on p. 40).

[Ath16] *Athens Wireless Metropolitan Network (AWMN)*. 2016. URL: `http://www.awmn.net/` (cit. on p. 1).

[BCF07] Moshe Babaioff, John Chuang, and Michal Feldman. "Incentives in peer-to-peer systems". In: *Algorithmic Game Theory*. Ed. by Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani. Cambridge University Press, 2007, pp. 593–612 (cit. on p. 7).

[BMT12] Ozalp Babaoglu, Moreno Marzolla, and Michele Tamburini. "Design and implementation of a P2P Cloud system". In: *27th Annual ACM Symposium on Applied Computing (SAC '12)*. New York, NY, USA: ACM Press, Mar. 2012, pp. 412–417 (cit. on p. 21).

[Bai+15a] Roger Baig, Felix Freitag, Amin M Khan, Agusti Moll, Leandro Navarro, Roger Pueyo Centelles, and Vladimir Vlassov. "Community Clouds at the Edge deployed in Guifi.net". In: *4th International Conference on Cloud Networking (CloudNet 2015)*. Niagara Falls, Canada: IEEE, Oct. 2015 (cit. on pp. 22, 83).

[Bai+15b] Roger Baig, Ramon Roca, Felix Freitag, and Leandro Navarro. "guifi.net, a crowdsourced network infrastructure held in common". In: *Computer Networks* 90 (Oct. 2015), pp. 150–165 (cit. on pp. 1, 2, 10, 30, 50, 52).

[Beb+09] Adam L. Beberg, Daniel L. Ensign, Guha Jayachandran, Siraj Khaliq, and Vijay S. Pande. "Folding@home: Lessons From Eight Years of Volunteer Distributed Computing". In: *8th IEEE International Workshop on High Performance Computational Biology (HiCOMB '09), within IPDPS*. Rome, Italy: IEEE, May 2009, pp. 1–8 (cit. on pp. 7, 21).

[BG06] Maria Bina and GM Giaglis. "Unwired Collective Action: Motivations of Wireless Community Participants". In: *International Conference on Mobile Business (ICMB'06)*. Copenhagen, Denmark: IEEE, June 2006, pp. 31–31 (cit. on p. 7).

[Bra+13] Bart Braem, Roger Baig Viñas, Aaron L. Kaplan, Axel Neumann, Ivan Vilata i Balaguer, Blaine Tatum, Malcolm Matson, Chris Blondia, Christoph Barz, Henning Rogge, Felix Freitag, Leandro Navarro, Joseph Bonicioli, Stavros Papathanasiou, and Pau Escrich. "A case for research with and on community networks". In: *ACM SIGCOMM Computer Communication Review* 43.3 (July 2013), pp. 68–73 (cit. on pp. 1, 22, 42).

[Buy13] Umit Cavus Buyuksahin. "On Incentive Mechanisms for Resource Sharing in Community Clouds". Master's thesis. Universitat Politècnica de Catalunya, 2013 (cit. on p. 48).

[Buy+02] Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger. "Economic models for resource management and scheduling in Grid computing". In: *Concurrency and Computation: Practice and Experience* 14.13-15 (Nov. 2002), pp. 1507–1542 (cit. on p. 15).

[BAV05] Rajkumar Buyya, David Abramson, and Srikumar Venugopal. "The Grid Economy". In: *Proceedings of the IEEE* 93.3 (Mar. 2005), pp. 698–714 (cit. on p. 10).

[Cap+09] Justin Cappos, Ivan Beschastnikh, Arvind Krishnamurthy, and Tom Anderson. "Seattle: a platform for educational cloud computing". In: *40th ACM Technical Symposium on Computer Science Education (SIGCSE 2009)*. Chattanooga, USA: ACM, Mar. 2009, pp. 111–115 (cit. on p. 21).

[Cat+14] Simon Caton, Christian Haas, Kyle Chard, Kris Bubendorfer, and Omer F. Rana. "A Social Compute Cloud: Allocating and Sharing Infrastructure Resources via Social Networks". In: *IEEE Transactions on Services Computing* 7.3 (July 2014), pp. 359–372 (cit. on pp. 9, 13–15, 21, 24).

[Cha+12] Kyle Chard, Kris Bubendorfer, Simon Caton, and Omer F. Rana. "Social Cloud Computing: A Vision for Socially Motivated Resource Sharing". In: *IEEE Transactions on Services Computing* 5.4 (Jan. 2012), pp. 551–563 (cit. on p. 21).

[Chu+03] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. "PlanetLab: An Overlay Testbed for Broad-Coverage Services". In: *ACM SIGCOMM Computer Communication Review* 33.3 (July 2003), pp. 3–12 (cit. on pp. 7, 21).

[Clo16] *Cloudy GNU/Linux Distribution*. 2016. URL: `http://cloudy.community` (cit. on pp. 25, 83).

[Com16a] *Community Cloud Testbed*. 2016. URL: `http://wiki.clommunity-project.eu/testbed:start` (cit. on pp. 25, 76).

[Com16b] *Community-Lab: Community Networks Testbed by the CONFINE Project*. 2016. URL: `http://community-lab.net/` (cit. on p. 40).

[CW12] Costas Courcoubetis and Richard Weber. "Economic Issues in Shared Infrastructures". In: *IEEE/ACM Transactions on Networking* 20.2 (Apr. 2012), pp. 594–608 (cit. on p. 10).

[DP12] Salvatore Distefano and Antonio Puliafito. "Cloud@Home: Toward a Volunteer Cloud". In: *IT Professional* 14.1 (Jan. 2012), pp. 27–31 (cit. on p. 21).

[EOS07] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords". In: *American Economic Review* 97.1 (2007), pp. 242–259 (cit. on p. 15).

[FK03] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier, 2003 (cit. on pp. 7, 10).

[Frei16] *Freifunk*. 2016. URL: `http://freifunk.net` (cit. on p. 1).

[Fre+14] Felix Freitag, Leila Sharifi, Amin M Khan, Leandro Navarro, Roger Baig, Pau Escrich, and Luis Veiga. "A Look at Energy Efficient System Opportunities with Community Network Clouds". In: *Workshop on Energy-Efficient System (EES), within 2nd International Conference on ICT for Sustainability (IST4S 2014)*. Stockholm, Sweden, Aug. 2014 (cit. on p. 83).

[Fun16] *FunkFeuer*. 2016. URL: `http://funkfeuer.at/` (cit. on p. 1).

[GC05] Daniel Grosu and Anthony T. Chronopoulos. "Noncooperative load balancing in distributed systems". In: *Journal of Parallel and Distributed Computing* 65.9 (Sept. 2005), pp. 1022–1034 (cit. on p. 9).

[GB14] Nikolay Grozev and Rajkumar Buyya. "Inter-Cloud architectures and application brokering: Taxonomy and survey". In: *Software: Practice and Experience* 44.3 (Mar. 2014), pp. 369–390 (cit. on p. 15).

[Gui+14] Yang Gui, Zhenzhe Zheng, Fan Wu, Xiaofeng Gao, and Guihai Chen. "SOAR: Strategy-proof auction mechanisms for distributed cloud bandwidth reservation". In: *IEEE International Conference on Communication Systems (ICCS 2014)*. Macau: IEEE, Nov. 2014, pp. 162–166 (cit. on p. 12).

[Gui16] *Guifi.net: Open, Free and Neutral Network Internet for everybody*. 2016. URL: `http://guifi.net` (cit. on p. 1).

[Guo+13] Jian Guo, Fangming Liu, Dan Zeng, John C S Lui, and Hai Jin. "A cooperative game based allocation for sharing data center networks". In: *32nd IEEE International Conference on Computer Communications (INFOCOM'13)*. Turin, Italy: IEEE, Apr. 2013, pp. 2139–2147 (cit. on pp. 12–14).

[HT04] J Halpern and V Teague. "Rational Secret Sharing and Multiparty Computation: Extended Abstract". In: *STOC*. 2004, pp. 623–632 (cit. on pp. 16, 17).

[Hur73] Leonid Hurwicz. "The Design of Mechanisms for Resource Allocation". In: *The American Economic Review* 63.2 (1973), pp. 1–30 (cit. on p. 9).

[Jan+11] R. Jana, Karthik N. Kannan, Yih-Farn Chen, R. Jana, and Karthik N. Kannan. "Using Generalized Second Price Auction for Congestion Pricing". In: *IEEE Global Telecommunications Conference (GLOBECOM 2011)*. Dec. 2011 (cit. on p. 54).

[Jan+14] Minsung Jang, Karsten Schwan, Ketan Bhardwaj, Ada Gavrilovska, and Adhyas Avasthi. "Personal clouds: Sharing and integrating networked resources to enhance end user experiences". In: *33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14)*. Toronto, Canada: IEEE, Apr. 2014, pp. 2220–2228 (cit. on p. 21).

[Jim+13] Javi Jiménez, Roger Baig, Pau Escrich, Amin M Khan, Felix Freitag, Leandro Navarro, Ermanno Pietrosemoli, Marco Zennaro, Amir H Payberah, and Vladimir Vlassov. "Supporting cloud deployment in the Guifi.net community network". In: *5th Global Information Infrastructure and Networking Symposium (GIIS 2013)*. Trento, Italy: IEEE, Oct. 2013 (cit. on p. 83).

[KBF15] Amin M Khan, Umit Cavus Buyuksahin, and Felix Freitag. "Incentive-based resource assignment and regulation for collaborative cloud services in community networks". In: *Journal of Computer and System Sciences* 81.8 (Dec. 2015), pp. 1479–1495 (cit. on pp. 33, 48).

[KBF14] Amin M Khan, Umit Cavus Buyuksahin, and Felix Freitag. "Prototyping Incentive-Based Resource Assignment for Clouds in Community Networks". In: *28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014)*. Victoria, Canada: IEEE, May 2014, pp. 719–726 (cit. on pp. 40, 48).

[KBF13] Amin M Khan, Umit Cavus Buyuksahin, and Felix Freitag. "Towards Incentive-based Resource Assignment and Regulation in Clouds for Community Networks". In: *Economics of Grids, Clouds, Systems, and Services*. Ed. by Jörn Altmann Altmann, Kurt Vanmechelen, and Omer F. Rana. Vol. 8193. Lecture Notes in Computer Science. Zaragoza, Spain: Springer International Publishing, Sept. 2013, pp. 197–211 (cit. on pp. 37, 48).

[KF14a] Amin M Khan and Felix Freitag. "Exploring the Role of Macroeconomic Mechanisms in Voluntary Resource Provisioning in Community Network Clouds". In: *11th International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2014)*. Vol. 290. Advances in Intelligent Systems and Computing. Salamanca, Spain: Springer International Publishing, June 2014, pp. 269–278 (cit. on p. 82).

[KF14b] Amin M Khan and Felix Freitag. "Sparks in the Fog: Social and Economic Mechanisms as Enablers for Community Network Clouds". In: *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* 3.8 (2014) (cit. on pp. 22, 82).

[Kha+15a] Amin M Khan, Felix Freitag, Smrati Gupta, Victor Muntés-Mulero, Jacek Dominiak, and Peter Matthews. "On Supporting Service Selection for Collaborative Multi-Cloud Ecosystems in Community Networks". In: *29th IEEE International Conference on Advanced Information Networking and Applications (AINA 2015)*. Gwangju, Korea, Mar. 2015 (cit. on p. 83).

[KFN16] Amin M Khan, Felix Freitag, and Leandro Navarro. "Community Clouds". In: *Encyclopedia of Cloud Computing*. Ed. by San Murugesan and Irena Bojanova. Wiley-IEEE, June 2016 (cit. on pp. 20, 26, 82).

[Kha+16a] Amin M Khan, Felix Freitag, Leandro Navarro, and Roger Baig. "Enabling Clouds in Community Networks". In: *European Project Space on Research and Applications of Information and Communication Systems*. Ed. by Carlos Cerqueira and James Uhomoibhi. Lisbon, Portugal: SCITEPRESS, 2016 (cit. on p. 83).

[KFR15] Amin M Khan, Felix Freitag, and Luis Rodrigues. "Current Trends and Future Directions in Community Edge Clouds". In: *4th International Conference on Cloud Networking (CloudNet 2015)*. Niagara Falls, Canada: IEEE, Oct. 2015 (cit. on pp. 21, 27, 82).

[KSF14] Amin M Khan, Mennan Selimi, and Felix Freitag. "Towards Distributed Architecture for Collaborative Cloud Services in Community Networks". In: *6th International Conference on Intelligent Networking and Collaborative Systems (INCoS 2014)*. Salerno, Italy: IEEE, Sept. 2014 (cit. on p. 27).

[Kha+13] Amin M Khan, Leila Sharifi, Leandro Navarro, and Luis Veiga. "Clouds of Small Things: Provisioning Infrastructure-as-a-Service from within Community Networks". In: *2nd International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2013), within IEEE WiMob*. Lyon, France: IEEE, Oct. 2013, pp. 16–21 (cit. on p. 83).

[Kha+16b] Amin M Khan, Xavier Vilaça, Luis Rodrigues, and Felix Freitag. "A Distributed Auctioneer for Resource Allocation in Decentralized Systems". In: *36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016)*. Nara, Japan, June 2016 (cit. on p. 79).

[Kha+15b] Amin M Khan, Xavier Vilaça, Luis Rodrigues, and Felix Freitag. "Towards Incentive-Compatible Pricing for Bandwidth Reservation in Community Network Clouds". In: *12th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2015)*. Cluj-Napoca, Romania: Springer International Publishing, Sept. 2015 (cit. on pp. 52, 79).

[KA06] Samee Ullah Khan and Ishfaq Ahmad. "Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation". In: *20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*. IEEE, 2006 (cit. on p. 15).

[Kri09] Vijay Krishna. *Auction Theory*. Academic Press, 2009, p. 336 (cit. on p. 15).

[Lai+04] Kevin Lai, Lars Rasmusson, Eytan Adar, Stephen Sorkin, Li Zhang, and Bernardo A. Huberman. "Tycoon: an Implementation of a Distributed, Market-based Resource Allocation System". In: *Multiagent and Grid Systems* 1.3 (Dec. 2004), pp. 169–182 (cit. on pp. 9, 10, 14, 15, 51).

[Lee+07] Seungjoon Lee, Dave Levin, Vijay Gopalakrishnan, and Bobby Bhattacharjee. "Backbone construction in selfish wireless networks". In: *ACM SIGMETRICS Performance Evaluation Review* 35.1 (2007), p. 121 (cit. on pp. 9, 10).

[Leo13] Xavier Leon. "Economic regulation for multi tenant infrastructures". PhD Thesis. Universitat Politècnica de Catalunya, 2013 (cit. on p. 10).

[Li+13] Hongxing Li, Chuan Wu, Zongpeng Li, and Francis C. M. Lau. "Profit-maximizing virtual machine trading in a federation of selfish clouds". In: *32nd IEEE International Conference on Computer Communications (INFOCOM'13)*. Turin, Italy: IEEE, Apr. 2013, pp. 25–29 (cit. on p. 14).

[LLZ15] Zongpeng Li, Baochun Li, and Yuefei Zhu. "Designing Truthful Spectrum Auctions for Multi-hop Secondary Networks". In: *IEEE Transactions on Mobile Computing* 14.2 (Feb. 2015), pp. 316–327 (cit. on p. 15).

[LAN03] Helger Lipmaa, N. Asokan, and Valtteri Niemi. "Secure Vickrey Auctions without Threshold Trust". In: *Financial Cryptography*. Ed. by Matt Blaze. Vol. 2357. LNCS 1. Springer Berlin Heidelberg, 2003, pp. 87–101 (cit. on p. 16).

[Liu+10] Zhengye Liu, Prithula Dhungel, Di Wu, Chao Zhang, and Keith W. Ross. "Understanding and Improving Ratio Incentives in Private Communities". In: *ICDCS*. IEEE, 2010, pp. 610–621 (cit. on pp. 9, 15).

[Luco0a] David Lucking-Reiley. "Auctions on the Internet: What's Being Auctioned, and How?" In: *The Journal of Industrial Economics* 48.3 (2000), pp. 227–252 (cit. on p. 14).

[Luco0b] David Lucking-Reiley. "Vickrey Auctions in Practice: From Nineteenth-Century Philately to Twenty-First-Century E-Commerce". In: *Journal of Economic Perspectives* 14.3 (2000), pp. 183–192 (cit. on pp. 14, 15).

[MT14] Patrick Maillé and Bruno Tuffin. *Telecommunication Network Economics: From Theory to Applications*. Cambridge University Press, 2014 (cit. on pp. 10, 52, 54, 58, 60, 61).

[MB09] Alexandros Marinos and Gerard Briscoe. "Community Cloud Computing". In: *1st International Conference on Cloud Computing (CloudCom 2009)*. Ed. by MartinGilje Jaatun, Gansen Zhao, and Chunming Rong. Vol. 5931. LNCS. Beijing, China: Springer Berlin Heidelberg, Dec. 2009, pp. 472–484 (cit. on pp. 19, 21).

[MG11] Peter Mell and Timothy Grance. "The NIST Definition of Cloud Computing". In: *NIST Special Publication* 800.145 (2011) (cit. on p. 20).

[MR14] Silvio Micali and Michael O. Rabin. "Cryptography Miracles, Secure Auctions, Matching Problem Verification". In: *Communications of the ACM* 57 (2014), pp. 85–93 (cit. on p. 16).

[MB16] San Murugesan and Irena Bojanova. *Encyclopedia of Cloud Computing*. Ed. by San Murugesan and Irena Bojanova. Wiley-IEEE, June 2016, p. 765 (cit. on p. 20).

[MS83] Roger B Myerson and Mark A Satterthwaite. "Efficient mechanisms for bilateral trading". In: *Journal of Economic Theory* 29.2 (Apr. 1983), pp. 265–281 (cit. on pp. 13, 62).

[Nin16] *Ninux.org Wireless Network Community*. 2016. URL: http://ninux.org (cit. on p. 1).

[NR99] Noam Nisan and Amir Ronen. "Algorithmic Mechanism Design". In: *STOC*. New York, NY, USA: ACM Press, Apr. 1999, pp. 129–140 (cit. on pp. 7, 9, 11, 14, 54, 60).

[NFL12] Di Niu, Chen Feng, and Baochun Li. "Pricing cloud bandwidth reservations under demand uncertainty". In: *SIGMETRICS*. New York, NY, USA: ACM Press, 2012, p. 151 (cit. on pp. 11, 12).

[NL13] Di Niu and Baochun Li. "An Efficient Distributed Algorithm for Resource Allocation in Large-Scale Coupled Systems". In: *IEEE INFOCOM*. Turin, Italy: IEEE, Apr. 2013, pp. 1501–1509 (cit. on p. 11).

[Niy13] Dusit Niyato. "Auction Approaches for Resource Allocation in Wireless Systems: A Survey". In: *IEEE Communications Surveys & Tutorials* 15.3 (2013), pp. 1020–1041 (cit. on p. 9).

[Nys12] *NYSE Technologies Introduces the World's First Capital Markets Community Platform*. 2012. URL: http://www1.nyse.com/press/1306838249812.html (cit. on p. 21).

[Ope16a] *OpenNebula: Open Source Data Center Virtualization*. 2016. URL: http://opennebula.org/ (cit. on p. 22).

[Ope16b] *OpenStack: Open Source Cloud Computing Software*. 2016. URL: http://www.openstack.org/ (cit. on p. 22).

[Ope16c] *OpenWrt: Linux distribution for embedded devices*. 2016. URL: http://openwrt.org/ (cit. on p. 41).

[Opt12] *Optum Health Care Cloud Environment*. 2012. URL: http://www.unitedhealthgroup.com/newsroom/articles/news/optum/2012/0214cloud.aspx (cit. on p. 21).

[Pal+13] Francesco Palmieri, Luigi Buonanno, Salvatore Venticinque, Rocco Aversa, and Beniamino Di Martino. "A distributed scheduling framework based on selfish autonomous agents for federated cloud environments". In: *Future Generation Computer Systems* 29.6 (Aug. 2013), pp. 1461–1472 (cit. on p. 14).

[Pico5] *Pico Peering Agreement v1.0*. 2005. URL: http://www.picopeer.net (cit. on pp. 8, 10, 30).

[Pop+12] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. "FairCloud: sharing the network in cloud computing". In: *SIGCOMM*. New York, NY, USA: ACM Press, 2012, p. 187 (cit. on pp. 11, 12).

[Pun+13] Magdalena Punceva, Ivan Rodero, Manish Parashar, Omer F. Rana, and Ioan Petri. "Incentivising resource sharing in social clouds". In: *Concurrency and Computation: Practice and Experience* (Mar. 2013) (cit. on p. 13).

[Rah+10] R. Rahman, M. Meulpolder, D. Hales, J. Pouwelse, D. Epema, and H. Sips. "Improving Efficiency and Fairness in P2P Systems with Effort-Based Incentives". In: *IEEE International Conference on Communications (ICC 2010)*. Cape Town, South Africa: IEEE, May 2010, pp. 1–5 (cit. on pp. 8, 29, 33).

[RS81] John G. Riley and William F Samuelson. "Optimal Auctions". In: *The American Economic Review* 71.3 (1981), pp. 381–392 (cit. on p. 9).

[RD10] Rodrigo Rodrigues and Peter Druschel. "Peer-to-peer systems". In: *Communications of the ACM* 53.10 (2010), pp. 72–82 (cit. on p. 7).

[Sano0] Tuomas Sandholm. "Issues in Computational Vickrey Auctions". In: *International Journal of Electronic Commerce* 4.3 (2000), pp. 1–35 (cit. on pp. 14, 16).

[Sat+09] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. "The Case for VM-Based Cloudlets in Mobile Computing". In: *IEEE Pervasive Computing* 8.4 (Oct. 2009), pp. 14–23 (cit. on p. 22).

[Sel+15] Mennan Selimi, Amin M Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. "Cloud services in the Guifi.net community network". In: *Computer Networks* 93.P2 (Dec. 2015), pp. 373–388 (cit. on pp. 25, 27, 83).

[Ser16] *Serf*. 2016. URL: `https://www.serfdom.io/` (cit. on p. 41).

[SL14] Haiying Shen and Zhuozhao Li. "New bandwidth sharing and pricing policies to achieve a win-win situation for cloud provider and tenants". In: *33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14)*. Toronto, Canada: IEEE, Apr. 2014, pp. 835–843 (cit. on pp. 12, 13).

[She+10] Xuemin Shen, Heather Yu, John Buford, and Mursalin Akon. *Handbook of Peer-to-Peer Networking*. Vol. 1. Springer Heidelberg, 2010 (cit. on pp. 7, 21, 33).

[Shi+14] W Shi, L Zhang, C Wu, Z Li, and F C M Lau. "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing". In: *SIGMETRICS*. 2014, pp. 71–83 (cit. on p. 11).

[TTL05] Douglas Thain, Todd Tannenbaum, and Miron Livny. "Distributed computing in practice: the Condor experience". In: *Concurrency and Computation: Practice and Experience* 17.2-4 (Feb. 2005), pp. 323–356 (cit. on p. 21).

[Veg15] Davide Vega D'Aurelio. "Incentives for sharing heterogeneous resources in distributed systems: a participatory approach". PhD thesis. Barcelona, Spain: Universitat Politècnica de Catalunya, 2015 (cit. on p. 8).

[Veg+12] Davide Vega, Llorenc Cerda-Alabern, Leandro Navarro, and Roc Meseguer. "Topology patterns of a community network: Guifi.net". In: *1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2012), within IEEE WiMob*. Barcelona, Spain: IEEE, Oct. 2012, pp. 612–619 (cit. on pp. 31, 32, 37).

[Veg+13] Davide Vega, Roc Messeguer, Sergio F. Ochoa, and Felix Freitag. "Sharing Hardware Resources in Heterogeneous Computer-Supported Collaboration Scenarios". In: *Integrated Computer-Aided Engineering* 20.1 (2013), pp. 59–77 (cit. on pp. 8, 29, 33).

[Wal+92] C.A. Waldspurger, T. Hogg, B.A. Huberman, J.O. Kephart, and W.S. Stornetta. "Spawn: a distributed computational economy". In: *IEEE Transactions on Software Engineering* 18.2 (1992), pp. 103–117 (cit. on p. 9).

[WRM12] Qian Wang, Kui Ren, and Xiaoqiao Meng. "When cloud meets eBay: Towards effective pricing for cloud computing". In: *IEEE INFOCOM*. Orlando, FL, USA: IEEE, Mar. 2012, pp. 936–944 (cit. on pp. 9, 11).

[Wil99] Uri Wilensky. *NetLogo*. 1999. URL: http://ccl.northwestern.edu/netlogo/ (cit. on p. 55).

[Wir10] *Wireless Commons License for Open, Free & Neutral Network (OFNN)*. 2010. URL: http://guifi.net/es/ProcomunXOLN (cit. on pp. 8, 10, 30).

[Xia+13] Yong Xiao, Jianwei Huang, Chau Yuen, and Luiz A. DaSilva. "Fairness and efficiency tradeoffs for user cooperation in distributed wireless networks". In: *INFOCOM*. Turin, Italy: IEEE, Apr. 2013, pp. 285–289 (cit. on pp. 9, 10).

[Yi+11] Sangho Yi, Emmanuel Jeannot, Derrick Kondo, and David P. Anderson. "Towards Real-Time, Volunteer Distributed Computing". In: *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011)*. Newport Beach, CA, USA: IEEE, May 2011, pp. 154–163 (cit. on p. 21).

[Zmq16] *ZeroMQ*. 2016. URL: http://zeromq.org/community (cit. on p. 76).

[Zha+13] Hong Zhang, Bo Li, Hongbo Jiang, Fangming Liu, Athanasios V Vasilakos, and Jiangchuan Liu. "A framework for truthful online auctions in cloud computing with heterogeneous user demands". In: *IEEE INFOCOM*. Turin, Italy: IEEE, Apr. 2013, pp. 1510–1518 (cit. on p. 11).

[ZAM10] K Zhang, Nick Antonopoulos, and Z Mahmood. "A taxonomy of incentive mechanisms in peer-to-peer systems: Design requirements and classification". In: *International Journal on Advances in Networks and Services* 3.1 (2010), pp. 196–205 (cit. on p. 7).

[ZLW14] Linquan Zhang, Zongpeng Li, and Chuan Wu. "Dynamic resource provisioning in cloud computing: A randomized auction approach". In: *33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14)*. Toronto, Canada: IEEE, Apr. 2014, pp. 433–441 (cit. on pp. 11, 12).

[Zha+15a] Xiaoxi Zhang, Zhiyi Huang, Chuan Wu, Zongpeng Li, and Francis C.M. Lau. "Online Auctions in IaaS Clouds: Welfare and Profit Maximization with Server Costs". In: *SIGMETRICS*. Vol. 43. 1. Portland, USA: ACM, June 2015, pp. 3–15 (cit. on p. 11).

[Zha+15b] Xiaoxi Zhang, Chuan Wu, Zongpeng Li, and Francis C M Lau. "A Truthful (1-ϵ)-Optimal Mechanism for On-demand Cloud Resource Provisioning". In: *INFOCOM*. 2015 (cit. on pp. 11, 51, 60, 74, 77).

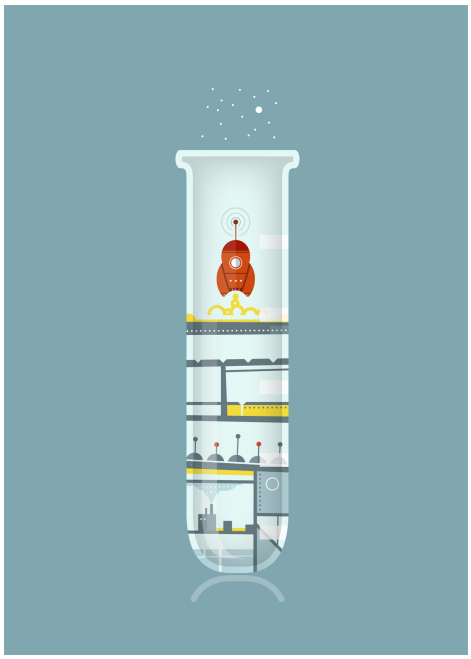
[ZLL14a] Han Zhao, Xinxin Liu, and Xiaolin Li. "Towards efficient and fair resource trading in community-based cloud computing". In: *Journal of Parallel and Distributed Computing* 74.11 (Aug. 2014), pp. 3087–3097 (cit. on p. 13).

[Zhe+15] Liang Zheng, Carlee Joe-Wong, Chee Wei Tan, Mung Chiang, and Xinyu Wang. "How to Bid the Cloud". In: *SIGCOMM*. New York, NY, USA: ACM Press, 2015, pp. 71–84 (cit. on p. 11).

[Zhe+14] Zhenzhe Zheng, Yang Gui, Fan Wu, and Guihai Chen. "STAR: Strategy-Proof Double Auctions for Multi-Cloud, Multi-Tenant Bandwidth Reservation". In: *IEEE Transactions on Computers* 64.7 (2014), pp. 2071–2083 (cit. on pp. 11, 12, 74, 76).

[ZLL14b] Haojie Zhou, Ka-Cheong Leung, and Victor O. K. Li. "Auction-based bandwidth allocation and scheduling in noncooperative wireless networks". In: *IEEE International Conference on Communications (ICC 2014)*. Sydney, Australia: IEEE, June 2014, pp. 2556–2561 (cit. on p. 10).

[Zho+08] Xia Zhou, Sorabh Gandhi, Subhash Suri, and Haitao Zheng. "eBay in the Sky: strategy-proof wireless spectrum auctions". In: *MobiCom*. New York, NY, USA: ACM Press, Sept. 2008 (cit. on pp. 9, 14, 15, 51).



This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Minion Pro, designed by Robert Slimbach in 1990 inspired by late Renaissance-era type and issued by Adobe in 2000. The headlines and captions are set in variations of Myriad Pro, a humanist sans-serif typeface designed by Robert Slimbach and Carol Twombly in 1990 and issued by Adobe in 2000. The above illustration was created by Ben Schlitter and released under CC BY-NC-ND 4.0. A template that can be used to format a PhD dissertation with this look & feel has been released under the permissive AGPL license, and can be found online at github.com/aminmkhan/Dissertate or from its lead author, Jordan Suchow, at suchow@post.harvard.edu.