A Cloudlet-based Multi-lateral Resource Exchange Framework for Mobile Users

Yu Wu and Lei Ying
School of Electrical, Computer and Energy Engineering
Arizona State University
Email: yuwu5@asu.edu, lei.ying.2@asu.edu

***Abstract*—The hardware improvement of mobile devices and pervasiveness of wireless technology expedite the convergence with the fast growing cloud computing trend, where the abundant resources on the cloud meet well with the deficiency of hand-held devices. *Cloudlet*, as a newly emerging paradigm "bringing the cloud closer" to the end users, features a more scalable deployment fashion where idle personal servers can be efficiently harnessed. Despite an envisioned monetary saving, such a paradigm confines itself to limited application scenarios, which fails to reach a wide realm of roaming users outside the distance coverage of the access points. In this paper, we propose a cloudlet-based multi-lateral resource exchange framework for mobile users, relying on no central entities. Inspired by the success of BitCoin, we design a novel virtual currency tailored for our framework. To realize an efficient resource exchange market, we also introduce flexible pricing strategies adopted by the individual users whom we assume are rational price-takers, with solid theoretical analysis on the equilibrium state and the stability. After elaborating the key functional modules, we introduce a prototype design enabling seamless trading among mobile users on Internet bandwidth as a proof-of-concept, with least user intervention. Both simulations and experiments are conducted to verify the practicality and efficiency of our system.**

I. Intrudoction

The proliferation of smartphones and the ubiquitousness of wireless coverage, echos the booming trends of cloud computing paradigm, with the abundant resources on the cloud complementing well to the deficiency of hand-held devices as agile services over the Internet. Coined as "mobile cloud computing" [1], a wealth of significant research and development have emerged from both academia and industry [2] [3]. From the perspective of cloud-side architecture, these efforts can be categorized into two schemes, *i.e.*, *central cloud* and *cloudlet*. *Central cloud* refers to a cloud system usually deployed by a single company or institution, *e.g.*, Amazon web services, Azure, *etc.*, and resources from a central cloud are usually only accessible via the Internet; *cloudlet* defines a more scalable cloud deployment paradigm with the goal of "bringing the cloud closer". Monetary costs can be saved by harnessing those otherwise idle personal servers of individual users, and the resources are accessible via WiFi if the mobile device is roaming within a distance from the hotspot.

However, neither of these two paradigms can lend themselves to a generic solution for a broader application scenario. For instance, the latency is of crucial importance for those time-sensitive applications, and the stringent turnaround time hinders the mobile devices from communicating with the distant cloud systems, *i.e.*, the central cloud. It's advised that WiFi accessible nearby cloudlets be deployed to avoid the routing delays, but the assumption is only valid in static environments, *e.g.*, home and offices, which precludes other application scenarios when users are on the move, where a central cloud is more preferable with its ubiquitous accessibility.

To unleash the real potential of mobile cloud computing and circumvent the problems mentioned above, we propose a distributed cloudlet ecosystem consisting of a collection of cloudlets which are organized into a peer-to-peer (P2P) overlay. The cloudlet servers are provisioned by the end users, usually the smartphone users, from either public or private clouds. [1] Mobile devices can directly connect to the nearby cloudlets via the low-latency one-hop WiFi network, and can also access the remote cloudlets with public IP addresses via the cellular channels. Moreover, service exchange is enabled among the community, realized in a peer-to-peer flavour, with differences lying in: (1) the classic "tit-for-tat" enforces a content-oriented exchange among peers. To cater for a much broader range of application models, a more generic exchange is desirable, *i.e.*, mobile devices can leverage CPU cycles, storage, broadband network and even software from the resource-rich cloudlet servers. Therefore, we define a generic mobile cloud computing framework, where "service/resource" can be exchanged rather than "contents" (In this paper, we will use service and resource interchangeably); (2) another limitation of the traditional TFT is, a peer can only benefit from the peers in the same swarm though she is interested in the services from another swarm, which will inevitably discourage her participation into such a resource-sharing system (As Fig. 1 shows, when mobile user A roams to a location near cloulet B, A can not leverage any service from B since cloudlet A is out of the reach of user B and no service exchange can happen). In contrast, our system allows multi-lateral resource exchanges among the users.

To realize such a system, several pending issues should be further addressed: (1) the system should not rely on a central entity but remain self-structured, for the sake of robustness; (2) the amount of each user's contribution should be quantified as a consensus by all the users to incentivize contributions and facilitate resource exchange. (3) the system should involve

[1] If from a public cloud, all user data should be encrypted.

least user intervention as a transparent service, since end users are not interested in the underlying complexities.

Our contributions in this work are: (1) we propose a generic cloudlet-based multi-lateral resource exchange framework for mobile users; (2) we introduce flexible resource provisioning schemes and the corresponding pricing strategies, which we believe also apply in other distributed systems; (3) based upon the framework we proposed, we also present a prototype system enabling Internet bandwidth leasing among mobile users.

The remainder of this paper is organized as follows. We discuss related work in Sec. II, and present the architecture of the framework in Sec. III. Sec. IV and Sec. V introduce the trading mechanism and the pricing strategy, respectively. A prototype specifically for Internet bandwidth leasing follows in Sec. VI, with Sec. VII discussing the evaluation results based on both simulations and the real-life prototype system. Sec. VIII concludes the paper and highlights the direction for our future work.

II. Related Work

Mobile Cloud Computing (MCC) [4] is embraced as one of the eruptive technologies in recent years, benefiting from both innovative development from the cloud computing paradigm and wireless technologies. A series of work [5] [6] [7] suggest offloading mobile devices' tasks to resource-rich cloud infrastructures. Zhang *et al.* [5] introduce an elastic mobile applicaton model by relocating part of the applications onto an IaaS cloud. Huang *et al.* [6] study the video transcoding problem by leveraging cloud resources as the SVC proxies.Wu *et al.* [7] present a mobile TV system with social functions, exploiting both a PaaS cloud and a IaaS cloud for video transcoding (CPU cycles), chunk buffering (storage) and traffic shaping (bandwidth). However, their solutions target at more specific scenarios which are difficult to be extended to a generic one. We instead advocate a generic mobile cloud computing framework where all types of underlying resources can be exposed to the mobile users through virtualization.. A further realization of a prototype system verifies the effectiveness of our framework.

Cloudlet, envisioned as a new MCC scheme, "brings the cloud to the mobile user" [8]. Satyanarayanan *et al.* [9] propose to offload computation workload on a mobile device to a nearby cloud system by dynamic VM synthesis. Kosta *et al.* [10] present a solid virtualization framework for code-level offloading from mobile devices to cloudlets. Work of this category manage to decrease the transmission delay and energy consumption which otherwise would adversely offset the performance gains achieved by offloading. However, the pure cloudlet architecture prohibits itself from adapting to the application scenarios where users are on the move. In contrast, our framework enables multi-lateral resource exchanges among different users where a roaming mobile user can flexibly access the resources supplied by other users' cloudlet servers. Sogata *et al.* [11] deliver a cloudlet-based architecture, *i.e.*, MOCHA, aiming to improve the response time for face recognition. One significant difference between their system and ours is, MOCHA still relies on the central cloud infrastructure while our system features a fully distributed architecture, relying on no central entities.

Along with the success of BitCoin, the academia has witnessed recent efforts, which further enforce the groundbreaking BitCoin-like decentralized currency system. Decker *et al.* study [12] how the information (exchange message, block announcement *etc.*) propagate in the Bitcoin network. There also exist some other work [13] [14] endeavouring to design "better" bitcoin-like currencies, with goals of strengthening the levels of security and anonymity. Yet, to our best knowledge, our system is the first work trying to seamlessly integrate the decentralized payment mechanism adopted by BitCoin into a fully distributed system with necessary tweaking.

III. Architecture

We advocate a market-driven system which has long been conceived as an effective decentralized mechanism with minimal communication overhead. A user earns the credits by selling services to other users, and the income can be used to purchase services from other cloudlets (In Fig. 1, user A may sell the "service" to user C while purchases the "service" from B). Budget-based systems have been well studied to mimic the marketplace in the physical world to enable multilateral exchanges, but a trusted authority is usually involved to either mediate the exchange, maintain the credit system or mint the currency, which ends up with a single point of control/failure, until the breakthrough of BitCoin. BitCoin, which has gained exponential popularity recently, is the first successful realizations of a fully distributed crypto-currency. Without any central authority in place, the peer-to-peer electronic currency system relies on digital signatures to prove ownership, and the entire transaction history is well maintained on each peer to prevent the double-spending issue. Although believed feasible to be incorporated to other systems as a payment mechanism, BitCoin is inappropriate to serve as a virtual currency in the cloudlet-based marketplace scenario mentioned above: (1) the total supply of coins is strictly capped *i.e.*, 21 millions, and the lack of currency supply will result in deflation and lack of circulation which hinder service sharing in a service-oriented platform. On the other hand, as bitcoins are more and more difficult to generate, it is unfair for late adopters who have to contribute much more resources/services to obtain the same amount of bitcoins. (2) the mining incentives are so strong, and miners continuously waste valuable computation powers which would contribute to the community otherwise. (3) as the price is soaring crazily (with a peak up to 1000 USD), the owners are more willing to hoard them instead of spending them, which would be disastrous for a system where active sharing is critical. Instead, we propose a novel currency system upon which our cloudlet resource exchange framework is built. For completeness, we briefly introduce the underlying mechanism of BitCoin adopted by our system and refer interested readers to [15] for the details.

Fig. 1. The limitation of bi-lateral resource exchange

Fig. 2 gives an overview of the architecture of our mobile cloudlet framework. A *Cloudlet* is deployed by each user to provide services to the other mobile users. Services can be registered and unregistered via the *Trackers* [2], where mobile users can retrieve the published services through look-up operations either by location or service type. Besides, like most real-world P2P systems, the *tracker* can also help an individual cloudlet server find other cloudlet servers to form peer-to-peer overlays to facilitate the distributed trading mechanism as will be introduced in Sec. IV. Service on a cloudlet is only accessible after payment, and the process is authenticated by the *Access Point*. The service leasing sequence diagram is shown in Fig. 3.

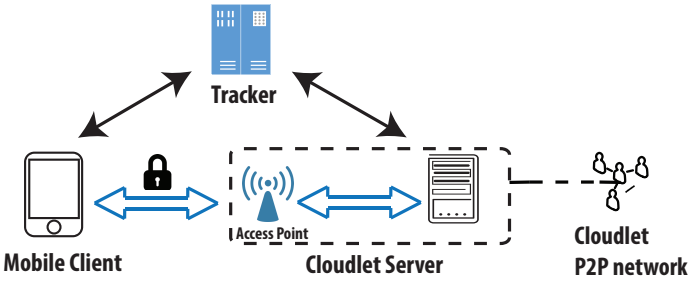

Fig. 2. The architecture of the interoperable cloudlet framework

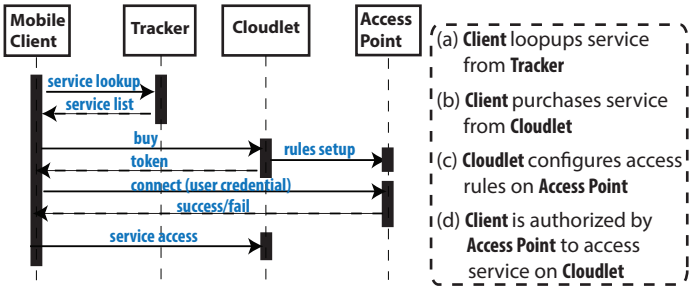

Fig. 3. The service leasing flow between mobile client and cloudlet

IV. Trading Mechanism

For the ease of implementation and trading, the available resources on a cloudlet server are split into multiple units, as shown in Fig. 4 (*e.g.*, resources like CPU and bandwidth can be efficiently split and provisioned via virtualization technologies [16] [17]). A unit resource is traded at a certain price, which will be automatically decided at the corresponding cloudlet's own discretion. The pricing strategy will be introduced later in Sec. V.

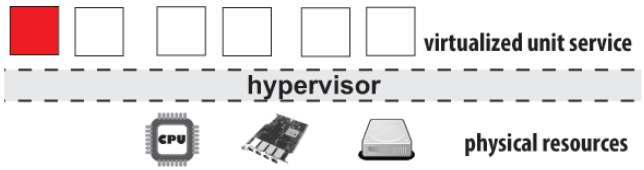

Fig. 4. Unit resources provisoned through virtualization

A. Transactions

Each unit resource is bound to a private unique random number which can trivially derive the corresponding public key via elliptic curve multiplication [18]. For security reason, the private random numbers are stored on the mobile device as the private keys. After a further hashing operation on the derived public key, the output result serves as the account address public to the market. Therefore, the budget system for each cloudlet contains a potentially unlimited accounts, with each account mapped to a unit resource. During the transaction of a certain unit resource, the cloudlet notifies the buyer of the corresponding account number, and the buyer may transfer the required budget to that account.

The virtual currency in circulation is implied in transactions, which simply conveys the fact that a certain amount of coins are migrated from one account to another in one trade. Thus, a transaction can be defined with multiple *inputs* and *outputs*. As shown in Fig. 6, each *output* specifies a payee account and the budget to transfer; each *input* references an *output* of a previous transaction dedicated to the same account. A valid transaction entails the total budget from all the *inputs* is enough to afford the expenditures indicated by all the *outputs*. A locking script is enforced on the output of one transaction specifying the condition to claim the currency in future. In our system, we require the account address of the payee is contained. An unlocking script is contained in one input of the future transaction to "solve" the locking script by providing the corresponding digital signatures signed by the corresponding private keys.

The process is depicted in Fig. 5. It's worthy to note that, the owner of the cloudlet does not intervene in this process, shielded from the underlying complexities.

B. Public Ledger

Once the transaction is established, it should be broadcast to the other cloudlets to confirm. All the confirmed transactions occurred within the same time period are encapsulated into one block. A block is organized into a public ledger in the chronological order by referencing the hash of the previous block, and each cloudlet maintains one copy of the ledger.

[2]Multiple trackers can be deployed to increase the scalability.

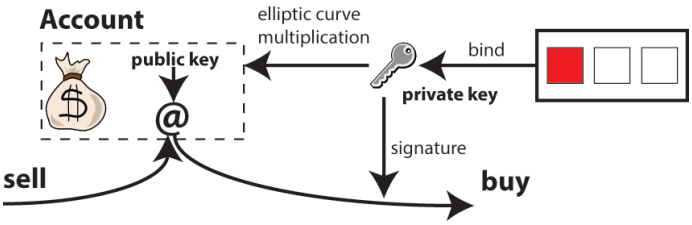

Fig. 5. The binding between a unit resource and an account number

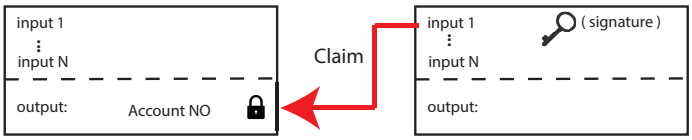

Fig. 6. The structure of transactions

Thus, the remaining budget of any account can be easily derived by checking the transaction histories against the ledger. On the other hand, the public ledger is of crucial importance to prevent the double-spending problems, as duplicate spending can be easily spotted by looking up the ledger for the past transactions.

The success of BitCoin-like coins lies largely in the individual confirmation process. Each cloudlet collects all the valid unconfirmed transactions into one *transaction pool* and tries to confirm them by solving a "puzzle" with dynamic non-trivial difficulties which approximately guarantees a new block is generated for a fixed interval. To be more specific, the puzzle is to "guess" a *nonce* value such that the hash value of the combination of the block header and the nonce should be below a target value. Therefore, we can see that the smaller the target value is, the more difficult the puzzle is.

Regardless of the difficulty, the result is extremely easy to verify. It features a reliable distributed system, as malicious users would pay prohibitively high costs to subvert the rule. If we assume a majority of users are honest, the security level of the system can be theoretically guaranteed, which also backs the success of BitCoin.

C. Currency LifeCycle

To incentivize users to reserve an amount of CPU cycles to maintain the public ledger, the system rewards the users who successfully generate new valid blocks an amount of coins. The difference from BitCoin which injects a fixed amount of currency (25 BTC as of writing) for each new block is, the amount of new coins minted in our system depends on the market activity, which, more specifically, equals to $\mathcal{C}_0 + \sigma(\mu^\rho \cdot \nu^\pi)$, where $\mathcal{C}_0$, ρ and π are constants, μ represents the total number of transactions in the new candidate block, ν represents the total transaction volumes (in terms of the amount of coins) incurred and σ is a monotonically increasing function. There exists a "coinbase" transaction inside each candidate block (a special input referencing no output of previous transactions), indicating to whom the newly minted currencies are allocated.

Assume each cloudlet spares an average of r hash rate, then the total system consisting of N_c cloudlets can generate a total of rN_c hash rates. If we assume the hash process follows a Poisson process, the expected number of hashes generated by the system is approximately $rN_c\Delta$ during each time period of Δ. If we further assume the range of hash values is $[h_{min}, h_{max}]$, the target value h_{target} can be estimated by solving $rN_c\Delta = \frac{h_{max}-h_{min}}{h_{target}-h_{min}}$, which can be used as the initial value. Due to user dynamics, the target value should be dynamically adjusted accordingly. If the actual time to generate a new block at t is $\Delta(t)$ with a target value $h_{target}(t)$, then the target at next period $h_{target}(t+1)$ is adjusted to $h_{target}(t) * \frac{\Delta(t)}{\Delta}$. The rationale is that, if the time to generate a new block is longer than expected, the difficulty should be lowered (increase h_{target}). Otherwise, the difficulty should be increased (reduce h_{target}).

To smooth the minting variance [19], our system also adopts a P2Pool-like [20] mechanism, allowing a certain number of cloudlets voluntarily forming a minting swarm to generate a new block together. In such a swarm, each cloudlet continuously compete to generate a "share" of the candidate block under a fixed targeted hash value H. In normal cases, $H > h_{target}$, so a "share" is easier to find than a block. Although a "share" makes no actual contribution to the public ledger, it conveys a proof of work from a cloudlet in the minting swarm. Whenever a newly-generated "share" is found, the cloudlet will broadcast it to the swarm, and the other cloudlets will include the new "share" [3] after validation, and try to generate the next "share". The process will terminate when the hash value of a "share" in the swarm is less than h_{target}, which will be immediately broadcast to the whole system as a new block. The minted coins will be allocated to all the cloudlets in the swarm, proportionally according to the number of "shares" each cloudlet generate in the new block, which is indicated by the "coinbase" transaction in the block. The whole process is illustrated in Fig. 7. In this example, if the newly minted "share" satisfies the global difficulty, cloudlet 5 will be rewarded with $\frac{2}{9}$ of the total minted coins.

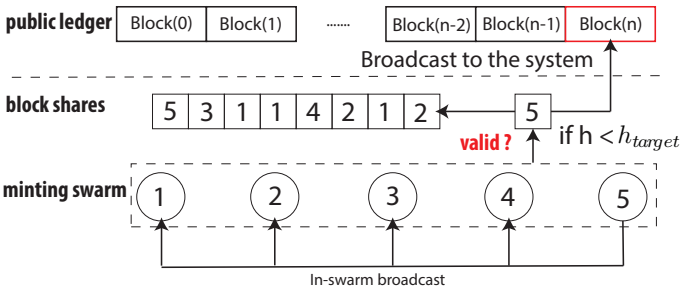

Fig. 7. The currency genertion process by a minting swarm

[3]add an extra output to the "coinbase" transaction, indicating the cloudlet contributing the "share' as a payee

To deal with the inflation effect of the system, the currency is devaluated by a percentile η for each time period. For instance, the value of the coins minted at time t_1, *i.e.*, $\$(t_1)$, will shrink to $\$(t_1) \cdot (1-\eta)^{t_2-t_1}$ at time t_2. In an equilibrium state where the expected amount of transactions is $\mathbb{E}(\mu)$ and the expected volume of transactions is $\mathbb{E}(\nu)$ for each period, the total wealth in the system approximately equals to $\frac{\mathcal{C}_0+\sigma(\mathbb{E}(\mu)^\rho \cdot \mathbb{E}(\nu)^\pi)}{\eta}$.

V. Pricing Strategy

A. Mechanism Design

We consider a system spanning across multiple locations, and the location is denoted as $\mathcal{S}$. Let $x_i(t) = \{x_i^1(t), x_i^2(t), \ldots, x_i^{|\mathcal{S}|}(t)\}$ denote the resource request vector variable, where $x_i^s(t) \geq 0$ represents the resource consumed by user i from location s at time t; Similarly, $y_i(t) = \{y_i^1(t), y_i^2(t), \ldots, y_i^{|\mathcal{S}|}(t)\}$ denotes the resource provision vector variable, where $y_i^s(t) \geq 0$ represents the resource provided by user i's cloudlet server deployed from location s at time t.

Assume the utility perceived by a user i over resources from different locations are independent[4], and let $\Theta_i(x_i(t))$ denote the utility function of user i. Following the common assumptions from existing literature [21], we assume that $\Theta_i^s(\cdot)$ is continuous, strictly concave and increasing over $x_i(t)$, if the resource at location s is accessible by user i. For completeness, we assume $\frac{\partial \Theta_i(x_i(t))}{\partial x_i^s(t)} \equiv 0$ if resources at location s are too far from user i to request; Let $\phi_i(y_i)$ denote the cost function of user i, and $\phi_i^s(\cdot)$ is continuous, strictly convex and increasing over y_i^s, if user i has deployed a cloudlet server at location s, and $\frac{\partial \phi_i(y_i)}{\partial y_i^s} \equiv 0$ otherwise. Without loss of generality, both $\Theta_i(\cdot)$ and $\phi_i(\cdot)$ are assumed to be differentiable.

During a period of time $[0, T]$, resource supply variables $y(t)$ and request variables $x(t)$ achieve the global optimality if they are solutions to the following optimization problem (Eqn. 1).

$$\begin{aligned} \max_{x(t)\geq 0, y(t)\geq 0} \quad & \int_0^T \sum_{i\in\mathcal{N}} (\Theta_i(x_i(t)) - \phi_i(y_i(t)))\, dt \qquad (1) \\ s.t. \quad & \sum_{s\in\mathcal{S}} x_i(t) \leq \sum_{s\in\mathcal{S}} y_i(t) \quad \forall i \in \mathcal{N}, \forall t \in [0,T] \end{aligned}$$

However, to realize a user-driven distributed system where the resources are under control of the end users, we consider a market-based resource allocation mechanism. Users provide and request the resources at their own discretion periodically, to maximize their own payoff function, *i.e.*, $\Theta_i(x_i(t)) - \phi_i(y_i(t))$.

Resources from the same location s at time t are associated with an identical price $p_s(t)$, with the price vector denoted as $p(t)$. We can see that only relative price ratio matters, and the ratio serves as a signal indicating the transient scarceness of resources at different locations, which further incentivizes users to deploy and contribute more scarce resources.

[4]In our work, we assume the utility function is time independent, but it can be straightforwardly extended to the time dependent case.

We consider a market where all users are price-takers and trying to maximize their individual payoffs by trading resources. Assume $w_i(t)$ is the budget saving of user i at time t, then the optimization problem confronted by user i is shown in Eqn. 2

$$\max_{x_i(t)\geq 0, y_i(t)\geq 0} \int_0^T \Theta_i(x_i(t)) - \phi_i(y_i(t))\, dt \qquad (2)$$

$$\begin{aligned} s.t.\ (i) \quad & w_i(t) = \int_0^t (w_i(t-1) + p(t)\cdot y_i(t) \\ & \qquad -p(t)\cdot x_i(t))\, dt \geq 0 \quad \forall t \in [1,T] \\ (ii) \quad & w_i(0) = 0 \end{aligned}$$

In a dynamic distributed system, the resource price $p(t)$ will change depending on the demand-supply relationship over time which a user i can not accurately predict. We instead consider a discrete slotted domain $t = 1, 2, \ldots, T$, where each user i tries to maximize her payoff by deciding her request and supply at the current time slot.

In a practical setting, our system requires each user should publish all the services they want to sell at the right beginning of the time interval t, and the services stay the same for the complete time interval, *e.g.*, 1 hour. Users may adjust their requests and be charged at a much shorter time scale defined as the "bidding period" in our work, *e.g.*, 10 minutes, which is reasonable for a user sojourning for a coffee to lease a nearby WiFi. Given the prices $p(t^-)$ and the budget saving $w_i(t^-)$ of user i at the beginning of time interval t, the optimization problem at time t can be revised to Eqn. 3. The optimum solution $y_i(t)^*$ can be used by user i to decide the amount of resource to publish (sell).

$$\begin{aligned} \max_{x_i(t)\geq 0, y_i(t)\geq 0} \quad & \Theta_i(x_i(t)) - \phi_i(y_i(t)) \qquad (3) \\ s.t. \quad & p(t^-)\cdot x_i(t) \leq p(t^-)\cdot y_i(t) + w_i(t^-) \end{aligned}$$

The objective function tries to maximize the payoff function of user i, and the only constraint requires that the total expenditure of user i to purchase the resources should not exceed the sum of income by selling its resources and the budget saving. Since $\Theta_i(\cdot)$ is concave, $\phi_i(\cdot)$ is convex, and the inequality constraint is linear, the problem is a convex problem with strong duality guarantee. A simple primal-dual algorithm can solve the problem efficiently. Interested users are referred to [22]. Let $e_i(t)$ denote the optimal solution $y_i(t)^*$, serving as the initial service endowment of user i which can be treated as a constant during the t-th time slot. If we assume a time interval is divided into M bidding periods, we next consider how each user adjusts the resource request flexibly during each bidding period, *i.e.*, $t, t+\frac{1}{M}, \ldots, t+1-\frac{1}{M}$, with the optimization problem formulated as Eqn. 4.

$$\max_{x_i(\tau)\geq 0} \ \Theta_i(x_i(\tau) \geq 0) \qquad (4)$$

$$s.t. \quad w_i(\tau) = \sum_{t}^{\tau}(w_i(\tau - \frac{1}{M}) + p(\tau)\cdot e_i(t) - p(\tau)\cdot x_i(\tau)) \geq 0$$

We use $D_i^s(p(\tau), e_i(t)), \tau \in [t, t+1)$ to represent the optimal solution (*i.e.*, $x_i(\tau)^*$) as the demand function of user i at location s. As the total available resource in each location is limited, there exist a series of implicit capacity constraints, *i.e.*, $\sum_{i\in\mathcal{N}} x_i^s(\tau) \leq \sum_{i\in\mathcal{N}} e_i^s(t) \quad \forall s \in \mathcal{S}$. The question now is: *can we find efficient prices to achieve an equilibrium which clears the market resources at each location while maximizing each user's utility?*

We first define the aggregate excess demand function $z(p(\tau)) = \{z_1(p(\tau)), z_2(p(\tau)), \ldots, z_{|\mathcal{S}|}(p(\tau))\}$ given price at time τ, *i.e.*, $p(\tau)$, where $z_s(p(\tau)) = \sum_{i\in\mathcal{N}}(D_i^s(p(\tau), e_i(t)) - e_i^s(t))$.

The system can achieve an equilibrium state if we can find a price vector p such that

$$z_s(p(\tau)) = \begin{cases} \leq 0 & \text{if} \quad p_s(\tau) = 0 \\ = 0 & \text{if} \quad p_s(\tau) > 0 \end{cases} \tag{5}$$

Eqn. 5 states the supply should equal to the demand in a location s unless the the resource is free. The $t\hat{a}tonnement$ process [23] is a simple pricing model to drive the economy towards the equilibrium, with the discrete form shown in Eqn. 6.

$$p_s(\tau + \frac{1}{M}) - p_s(\tau) = \xi_s(p_s(\tau), z_s(p(\tau))) \tag{6}$$

$\xi_s(p_s, z_s(\cdot))$ is a sign-preserving function satisfying $\xi_s(p_s, 0) = 0$. The rationale behind is simple: if the aggregate demand at location s exceeds the total supply, *i.e.*, $z_s(p) > 0$, the seller may increase the unit resource price at s; otherwise if $z_s(p) < 0$, the unit price should be reduced. The process continues until $z(p) = 0$. The $t\hat{a}tonnement$ process can globally converge to a Walrasian Equilibrium [24] under the weak *gross substitute property* condition [25]:

$$\frac{\partial z_s(p)}{\partial p_{s'}} \geq 0 \quad \forall s' \neq s.$$

We can easily verify that our system model has the property, but we should prove at least one Walrasian Equilibrium exits since the weak *gross substitute property* does not promise the existence of a Walransian Equilibrium. Before the proof, the pricing mechanism of our system needs further normalization.

Price adjustment in a classical $t\hat{a}tonnement$ process does not enforce any restrictions on the bounds of the price dynamics, which may result in an ill-defined system with the resource price out of control. Furthermore, we can easily verify the demand functions are *homogeneous of degree zero*, since if p^* is an equilibrium price, $\alpha \cdot p^* \quad (\alpha > 0)$ is also an equilibrium price vector. Consider any neighbourhood of any equilibrium price p^*, *i.e.*, $\{p : ||p - p^*|| \leq \delta, \delta > 0\}$, the range will cover other different collinear equilibrium prices, suffering from local instability. To tackle the erratic price dynamics and maintain the prices at a reasonable level when the system scales, we artificially require that $\sum_{s\in\mathcal{S}} p_s = |\mathcal{S}|$, *i.e.*, the sum of the prices in the system equals to the number of locations. When ξ_s is defined as $\xi_s = K \cdot p_s(t) \cdot z_s(p(t))$ where K is a small enough step size. Theorem 5.1 guarantees if the initial prices are set appropriately (a simple choice is to set the initial price to 1 for each location), the prices can be dynamically adjusted within the restricted range, with the detailed proof in [22]. Besides the benefit of stabilizing and scaling the system, the price normalization is also one crucial step for the existence issue (Theorem 5.2).

Theorem 5.1: Under the definition of $\xi_s = K \cdot p_s(t) \cdot z_s(p(t))$,$\forall t > 0$, $\sum_{s\in\mathcal{S}} p_s(t+1) = \sum_{s\in\mathcal{S}} p_s(t)$

Theorem 5.2: There exists at least one Walrasian Equilibrium in the system.

The detailed proof will be shown in [22].

Therefore, Alg. 1 gives the corresponding practical algorithm executed by each cloudlet at time interval T to decide the optimal resource to register and the dynamic price strategy.

Algorithm 1 Service registration and dynamic pricing at time interval t

Input: initial price vector p_0.

1: Calculate the optimal resource to publish
2: Handle requests from mobile clients
3: **for** $\tau \in t + \frac{1}{M} \ldots t + 1 - \frac{1}{M}$ **do**
4: update price according to Eqn. 6;
5: **end for**
6: **Goto** Step 1.

B. Undesirable oscillation

The potential existence of multiple Walrasian Equilibria implies there is no guarantee upon the convergence of Alg. 1 to an optimal equilibrium or any near-optimal one , without a centralized entity in place. But it is still desirable if the trading process in any time interval converges to any one equilibrium. However, regarding to adjacent time intervals, we should not neglect the potential high dynamics, as the ending price (probably in an equilibrium state) of the preceding interval will impact the trading process of the subsequent interval, serving as the initial price based on which users will decide the quantities of resources to register. Therefore, erratic price oscillation may happen and the root cause is that users may constantly overcompensate for the imbalances in the preceding time interval, as will be shown in Sec. VII.

We next study how to control the price oscillation. If we change resource price at location s and fix the prices at the other locations, we can easily see $\frac{\partial z_s(p_{s,-s})}{\partial p_s} < 0$, in other words, $\forall\, p_{s,-s}^{'}, p_{s,-s}^{''}$, we have $(z_s(p_{s,-s}^{'}) - z_s(p_{s,-s}^{''}))(p_{s,-s}^{'} - p_{s,-s}^{''}) < 0$. We extend this strict monotonicity to all the locations in a trivially stronger expression (Eqn. 7), following our normalized pricing rules in Sec. 5.1.

$$\langle z(p(t)^{'}) - z(p(t)^{''}), p(t)^{'} - p(t)^{''} \rangle < 0, \forall p(t)^{'}, p(t)^{''} \tag{7}$$

where $\langle , \rangle$ is the inner product operation for vectors. We can further translate Eqn. 7 into Corollary 5.3.

Corollary 5.3: $\exists \beta > 0$, such that $\langle z(p(t)^{'}) - z(p(t)^{''}), p(t)^{'} - p(t)^{''} \rangle \leq -\beta ||p(t)^{'} - p(t)^{''}||^2$.

Proof: We know that $\langle p(t)^{'} - p(t)^{''}, p(t)^{'} - p(t)^{''} \rangle = ||p(t)^{'} - p(t)^{''}||^2 \geq 0$. Thus, $\exists \beta > 0$, such that $\langle z(p(t)^{'}) - z(p(t)^{''}), p(t)^{'} - p(t)^{''} \rangle + \beta \cdot \langle p(t)^{'} - p(t)^{''}, p(t)^{'} - p(t)^{''} \rangle \leq 0$. ■

Based on this assumption, Thm. 5.4 guarantees that if the variance of resource supply at adjacent time intervals is bounded, the dynamics of equilibrium prices are also bounded. For real world implementation, the system can enforce the resource supply variance of any cloudlet between adjacent time intervals within a percentile.

Theorem 5.4: Assume $e(t)$ and $p^*(t)$ are the resource supply and equilibrium price at t-th time interval, respectively. We have $||p(t+1)^* - p(t)^*|| \leq \gamma \cdot ||e(t+1) - e(t)||$, where $\gamma > 0$.

To prove Thm 5.4, we also have Lemma 5.5 and Lemma 5.6. For detailed proof, please refer to [22].

Lemma 5.5: $||z(p, t+1) - z(p, t)|| \leq A \cdot ||e(t+1) - e(t)||$.

, where $A > 0$.

Lemma 5.6: $||p(t)^* - p(t+1)^*|| \leq B \cdot ||z(p(t)^*, t+1) - z(p(t)^*, t)||$

, where $B > 0$.

VI. Prototype

Based upon the cloudlet-based resource exchange framework we have proposed, we have also implemented a prototype enabling Internet leasing among mobile users through their cloudlet servers. We seek to design a system by hiding all market complexities from users. To guarantee satisfied energy efficiency, we follow the design philosophy of splitting those energy-consuming tasks to the cloudlet side, while keeping necessary user-interactive tasks on the mobile side.

A. Key modules on mobile device

To facilitate easy-to-access operations for users, only necessary interfaces are exposed, *i.e.*, (a) service lookups, (b) payment, (c) budget query and (d) cloudlet management, while all market-related complexities are transparent to users. Fig. 8 presents the key modules deployed on mobile device:

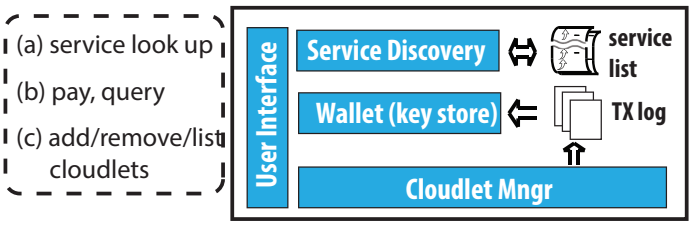

Fig. 8. Key modules on Mobile Device

▷ **Service Discovery** lookups Internet services by contacting trackers based upon user's preference, *e.g.*, location. Trackers respond the lookup requests with a list of available services, including the name (SSID), location, price, and quality.

▷ **Wallet** has the exclusive access to the private keys for each account (corresponding to each unit resource), and is responsible for trading with the cloudlet servers by signing transactions. To avoid overwhelming the limited storage of the mobile device, only transactions involving the user's owned cloudlets are locally stored as the *TX log*.

▷ **Cloudlet Mngr** manages all the cloudlet servers on behalf of the mobile users, via which users can add, remove and list owned cloudlets.

B. Key modules on cloudlet server

Mobile clients can only access the resources on the cloudlet servers after $802.1x$ authentication. A cloudlet consists of a server and an attached Radius-enabled [26] wireless router [5].

▷ **Trading** handles the requests from mobile clients. Once the payment is successful, a *token* (username-password pair) is returned to the user as the login credential. Accordingly, the *user profile* database is updated and the user can access the Internet service purchased until the specific *token* expires, whose entry will be deleted from the *user profile* database.

▷ **Service Mngr** dynamically calculates the amount of bandwidth to register, and decides the corresponding prices (Sec. V).

▷ **Hypervisor** virtualizes the physical bandwidth into a set of queues/pipes, with each affording up to a unit bandwidth. Multiple queues may be reserved for one mobile client, depending on the amount of unit bandwidth the user purchased. The module is realized through Dummynet [27].

▷ **Multiplexer** multiplexes traffics into the corresponding transmission queues based on the MAC address of the connected mobile devices, which are acquired when users login with their assigned credential.

▷ **Access Control** authenticates mobile clients' access to the cloudlet resource by checking against the *user profile* database. If successful, a secure channel is established, which is isolated from other channels when multiple mobile clients connect. The proxy is also configured on the device via the proxy auto-config (PAC) [28] script without users' intervention.

▷ **TX Mngr** maintains the *public ledger* and *transaction pool*. We implement this module based on Libbitcoin [29] (Sec. IV-A, Sec. IV-B).

▷ **Mint** reserves a certain amount of CPU cycles to confirm transactions in the system, by joining a minting swarm. (Sec. IV-C)

▷ **Syncer** periodically filters out all the transactions involving the cloudlet server, and transmits them to the corresponding mobile user for payment and balance query purposes.

Fig. 9 presents the key modules of the cloudlet.

[5]Radius is nowadays supported by regular off-the-shelf routers

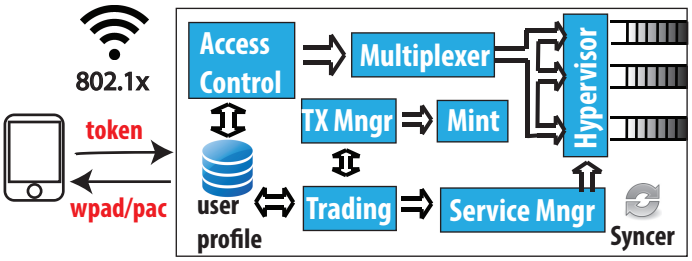

Fig. 9. Key modules on Cloudlet Server

VII. Performance Evaluation

We verify our market-driven pricing strategies and evaluate the performance of our system design, based on both large-scale simulations and a real-life prototype system. The prototype is realized with Ubuntu 14.04 LTS and an Airport Express (a low-end Radius-enabled wireless router manufactured by Apple Inc.). We implement the mobile logics on Google Nexus 5.

A. Experimental Settings

We consider a 800 user set roaming among 10 different locations, with each user possessing only 1 cloudlet server at a fixed location and assume: (1) 40% and 30% mobile users crowd into location 1 and 2, respectively, while the other locations accommodate an equal share of the remaining users; (2) all the cloudlet servers are uniformly distributed; (3) each time interval equals to 1 hour, and the bidding period is 10 minutes; (4) the unit bandwidth is set to 512 Kbps; (5) the size of the minting swarm is limited to 20. In addition, we select x^m as the utility functions and e^{n*y} as the cost functions, with m uniformly distributed among $[0.4, 0.41]$ and n uniformly distributed among $[1.04, 1.1]$. For currency minting, we set $\mathcal{C}_0 = 5$, $\sigma(\cdot) = log(\cdot)$, $\rho = 0.02$, $\pi = 0.01$ and the devaluation factor $\eta = 0.01$.

We run the system for 80 hours, *i.e.*, 80 time intervals.

B. Efficiency of the Pricing Strategy

We randomly select one cloudlet server (id 230) from the crowded location 1. Fig. 10 plots the unit resource price evolution over time. The curve labelled as "controlled" represents the pricing strategy enforced by restricting the variance percentile of the bandwidth registered of one cloudlet server between adjacent time intervals under 10% (Sec. V-B), and "uncontrolled" represents the case without such enforcement. We can see, the price variance in "controlled" strategy is limited and the equilibrium state can be achieved. On the other hand, prices between adjacent time intervals vibrate erratically. Fig. 11 plots the payoff value for the corresponding user over time and an interesting phenomenon is revealed: "controlled" pricing strategy efficiently directs the user to gradually improve her payoff function while "uncontrolled" fails due to instability of the system, resulting from the price dynamics.

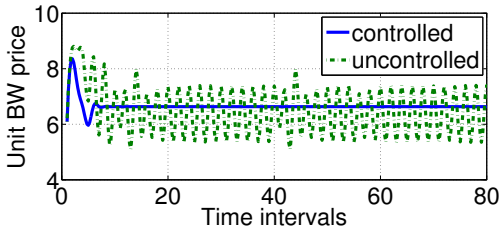

Fig. 10. Equilibrium price of unit bandwidth over time

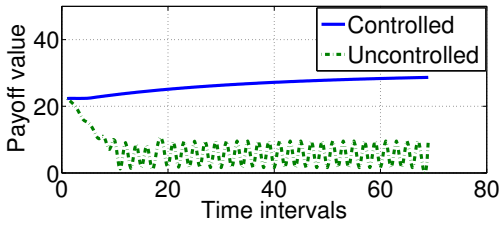

Fig. 11. Payoff value over time

C. Block generation

Fig. 12 shows the dynamics of the intervals between adjacent block generations. We can see the average time to find a new block by the system approximately equals to 10 minutes, *i.e.*, the bidding period.

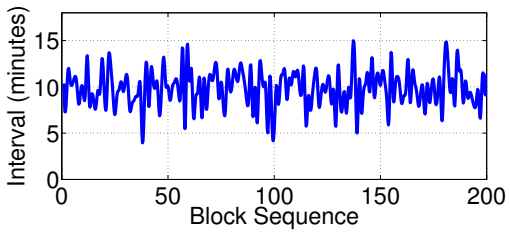

Fig. 12. Bandwidth over time

D. Dynamics of the social wealth

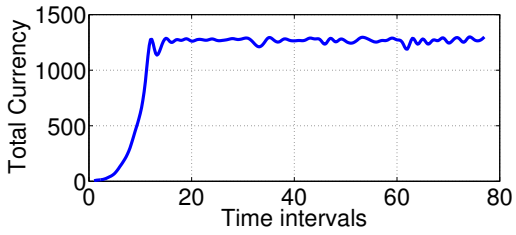

Fig. 13. Wealth over time

We observe the evolution of the social wealth over the time, and Fig. 13 plots the social wealth by calculating the overall currencies in circulation for each time interval. We can see the system needs a short "warm-up" time to gather enough coins for active resource trading. Then, the total wealth in the system reaches an equilibrium state, consistent with our theoretical analysis in Sec. IV-C.

E. Flexibility of the Bandwidth Leasing

To verify the the flexibility of the bandwidth leasing, we access a YouTube video via the mobile device by leasing the Internet bandwidth from our cloudlet prototype. Fig. 14 plots the bandwidth variation when 1 unit bandwidth ($512Kbps$) and 3 unit bandwidths ($1536Kbps$) are leased, respectively. The bandwidth statistics are sampled by each second. We can see different qualities of service are achieved. When 1 unit bandwidth is leased, a 240p video quality is streamed to the mobile device, while an up to 480p video playback can be achieved when 3 unit bandwidths are leased by the client. We can also observe that the actual peak bandwidth exceeds the bandwidth purchased from time to time, we believe it is caused either by the imprecise sampling method or the kernel-based traffic shaping solution adopted by Dummynet. The fact implies that it makes more sense to put the bandwidth multiplexer module to the *access point*, realized in reliable hardware solution. This will be part of our future work.

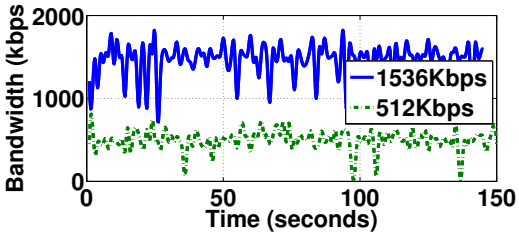

Fig. 14. Bandwidth over time

VIII. Conclusion and Future work

This paper proposes our design of a generic market-driven resource exchange framework for mobile users via their cloudlet servers. Inspired by the success of BitCoin, we introduce a new virtual currency system tailored for our framework. We also present in details the trading mechanism and the pricing strategy which we believe also apply in other distributed systems. To examine the effectiveness of the framework, we also implement a prototype enabling Internet bandwidth leasing among mobile users. Experiments with realistic settings verify the practicality and flexibility of the system. In our future work, we seek to extend the framework by enforcing more practical combinatorial restrictions. For instance, the storage service is inaccessible if no bandwidth is provided. In other words, resources should be traded in a well-defined bundle instead of independently.

Acknowledgement

This work was supported by the National Science Foundation under Grants CNS-1264012 and CNS-1255425, and the Defense Threat Reduction Agency (DTRA) under Grant HDTRA1-13-1-0030.

References

[1] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 369–392, 2014.

[2] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *Wireless Communications, IEEE*, vol. 20, no. 3, 2013.

[3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[4] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Middleware 2009*. Springer, 2009, pp. 83–102.

[5] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011.

[6] Z. Huang, C. Mei, L. Li, and T. Woo, "Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 201–205.

[7] Y. Wu, Z. Zhang, C. Wu, Z. Li, and F. C. Lau, "Cloudmov: Cloud-based mobile social tv," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 821–832, 2013.

[8] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*. ACM, 2012, pp. 29–36.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[10] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 945–953.

[11] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Computers and Communications (ISCC), 2012 IEEE Symposium on*. IEEE, 2012, pp. 000 059–000 066.

[12] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.

[13] S. Barber, X. Boyen, E. Shi, and E. Uzun, "Bitter to betterhow to make bitcoin a better currency," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 399–414.

[14] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 397–411.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, vol. 1, p. 2012, 2008.

[16] S. Rixner, "Network virtualization: Breaking the performance barrier," *Queue*, vol. 6, no. 1, p. 37, 2008.

[17] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Oct. 2003.

[18] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[19] *Variance: The Bitcoin Boogeyman that might just destroy the network.*, http://www.followthecoin.com/variance-bitcoin-boogeyman-might-just-destroy-network/.

[20] *P2Pool*, https://en.bitcoin.it/wiki/P2Pool.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.

[22] *A Cloudlet-based Multi-lateral Resource Exchange Framework for Mobile Users*, https://www.dropbox.com/s/tzt6dtrdxfd7es9/techreport.pdf.

[23] L. Walras, *Elements of pure economics*. Routledge, 2013.

[24] S. Bowles and H. Gintis, "Walrasian economics in retrospect," *Quarterly Journal of Economics*, pp. 1411–1439, 2000.

[25] T. Alpcan, H. Boche, M. L. Honig, and H. V. Poor, *Mechanisms and Games for Dynamic Spectrum Allocation*. Cambridge University Press, 2013.

[26] *Remote Authentication Dial In User Service (RADIUS)*, http://tools.ietf.org/html/rfc2865.

[27] *The dummynet project*, http://info.iet.unipi.it/~luigi/dummynet/.

[28] I. Cooper, I. Melve, and G. Tomlinson, "Internet web replication and caching taxonomy," 2001.

[29] *libbitcoin - Asynchronous C++ Bitcoin library*, http://libbitcoin.dyne.org/.