Market Design & Analysis for a P2P Backup System

Sven Seuken*
School of Engineering & Applied Sciences
Harvard University, Cambridge, MA
seuken@eecs.harvard.edu

Denis Charles, Max Chickering, Sidd Puri
Microsoft
One Microsoft Way, Redmond, WA
{cdx,dmax,siddpuri}@microsoft.com

ABSTRACT

In this paper we take the problem of a market-based P2P backup application and carry it through market design, to implementation, to theoretical and experimental analysis. While the long-term goal is an open market using real money, here we consider a system where monetary transfers are prohibited. We first describe the design of the P2P resource exchange market and the UI we developed. Second, we prove theorems on equilibrium existence and uniqueness. Third, we prove a surprising impossibility result regarding the limited controllability of the equilibrium and show how to address this. Fourth, we present a price update algorithm that uses daily supply and demand information to move prices towards the equilibrium and we provide a theoretical and experimental convergence analysis. The market design described in this paper is already implemented as part of a Microsoft research project on P2P backup systems and an alpha version of the software has been successfully tested.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Design, Economics

Keywords

Market Design, P2P, Online Backup, Exchange Market

1. INTRODUCTION: P2P BACKUP

With the increasing importance of information technology in our lives, we are also more and more dependent on being able to readily access all of our data all the time. However, users regularly lose valuable data because their hard drives crash, their laptops are stolen, etc. Already in 2003, the annual costs of data loss in the US was estimated to be $18.2 Billion.[1] With broadband connections becoming faster and cheaper, *online backup systems* are becoming increasingly attractive alternatives to traditional backup. There are hundreds of companies offering online backup services, e.g., SkyDrive, Idrive, Amazon S3. Most of these companies offer some storage for free and charge fees when the free quota is exceeded. All of these services, however, rely on large data centers and thus incur immense costs. The motivation for peer-to-peer (P2P) backup systems is that idle resources on the computers of millions of users can be used to avoid these costs. While the total network traffic increases with a P2P solution, the primary cost factors that can be avoided are 1) costs for hard drives, 2) energy costs for building, running and cooling data centers[2], 3) costs for large peak bandwidth usage, 4) personnel costs for computer maintenance. In a P2P backup system, these costs almost disappear because we can use lots of otherwise idle resources.

The main idea of P2P backup is that users provide some of their resources (storage space, upload bandwidth, download bandwidth, and online time) in exchange for using the backup service. A study performed by Microsoft in 2008 showed that about 40% of Windows users have more than half of their hard disk free and thus could be excellent candidates for using a P2P backup system. The rapidly decreasing costs for large hard drives might make P2P backup even more attractive in the future. In our own recent study [13], we showed that many users are not willing to pay the high fees for server-based backup and about half our participants said they would consider using P2P backup instead. Thus, there is definitely potential for P2P backup applications.

1.1 The Hidden Market Approach

Our P2P backup system is novel in that it uses a market to allocate resources, more efficiently than a non-market-based system could. We have implemented and successfully tested the system in alpha version. During the design phase for this system we followed a new paradigm that we recently introduced called "Hidden Market Design" [14]. The goal of a hidden market is to hide some of the complexities of the market from the user and to make the interaction for the user as seamless as possible. A P2P backup application is a particularly good example for hidden market design because the application targets millions of technically unsophisticated

*Part of this work was done while the author was a research intern at Microsoft.

[1] http://gbr.pepperdine.edu/033/dataloss.html

[2] In 2008, data centers in the US were responsible for about 3% of the country's energy consumption. Note that a P2P backup system cannot only reduce costs but is also more environmentally friendly due to reduced carbon emissions.

EC'10, June 7–11, 2010, Cambridge, Massachusetts, USA.

users, in a domain where markets/trading/currency etc. are very unnatural. As a result of this hidden markets approach, our resulting design is very unusual in that we provide our users with an indirect way to express their preferences.

Each resource in the market has a price, and the relative prices reflect the scarcity of the resources. The market design addresses the combinatorial nature of resources, namely that all users must provide a certain amount of all resources, even if they currently only consume a subset of them. For example, a user who only contributes space is useless to the system because no files could ever be sent or received from that user if no bandwidth is provided. The design must also allow users to express idiosyncratic preferences regarding how much of each resource they want to supply. Some users might need their own disk space a lot and prefer to sacrifice their internet connection. Other users might use their bandwidth for services like VOIP or file-sharing and might have a high disutility if the quality of those services were affected. We allow different users to provide different ratios of their resources, and we update prices regularly taking into account aggregate supply and demand.

1.2 Overview of Results

In this paper we present our market design for a P2P backup system and provide a theoretical and experimental analysis of its properties. At all times, the design and analysis was done for the actual implemented system. Our approach is novel and different from related work in that we follow the hidden market design paradigm [14] and use a very indirect market model. In our system, users do not continuously specify demand and supply vectors but instead only periodically choose bounds on their future maximum supply and demand, which makes the system much more usable.

After introducing an economic model, we define a *safety property* that shall guarantee that the system can always satisfy new incoming demand. Then, we define a *buffer equilibrium*, which is an equilibrium defined on supply and demand bounds. We prove that this equilibrium exists, is unique, and satisfies the safety property. However, we also show that the equilibrium cannot be easily controlled, in particular the size of the supply side buffer is out of the market operator's control. We explain the origin of this counter-intuitive result and show which design changes would be necessary to give the market operator more control. A price-update algorithm is introduced that takes system-wide supply and demand information and updates prices to drive the market towards the buffer equilibrium. Analytical and experimental analysis shows reasonable convergence speed when the initial price vector is chosen close enough to the equilibrium.

1.3 Related Work

In recent years there has been much research on P2P storage systems, electronic markets, distributed accounting, resource exchange systems, etc. Almost 10 years ago, the research projects OceanStore [8] and FarSite [3] already investigated the potential of distributed file systems using P2P. Both projects, however, did not do any kind of market design. More recently, researchers have looked at the incentive problem, often with the primary goal to enforce fairness (you get as much as you give). Samsara [5] is an accounting scheme that allows for fairness enforcement. In contrast to our design, their scheme is fully distributed. While such a design has some advantages, it prevents the use of sophisticated pricing and payment mechanisms that we employ.

The idea to use electronic markets for the efficient allocation of resources is even older than ideas regarding P2P storage systems. Already in 1996, Ygge et al. [16] proposed the use of computational markets for efficient power load management. More recently, grid networks and their efficient utilization have gotten more attention [9]. Fundamental to these designs is that participants are sophisticated users able to specify bids in an auction-like framework. While this assumption seems reasonable in energy markets or computational grid networks, we are targeting millions of users with our backup service and thus we cannot assume that users are able to directly act as traders on an exchange market.

The two papers most similar to our work are by Aperjis et al. [2] and Freedman et al. [7]. They analyze the potential of exchange economies for improving the efficiency of file-sharing networks. While the domain is similar to ours, the particular challenges they face are quite different. They use a market to balance supply and demand with respect to popular or unpopular files. However, in their domain there is only one scarce resource, namely upload bandwidth, while we must design an exchange market for multiple resources.

There exist multiple P2P backup applications that are being used in practice. The application most similar to ours is *Wuala* (`www.wuala.com`). However, none of these systems are market-based and do not allow different users to provide different resource ratios. Thus, these systems exhibit economic inefficiencies compared to our exchange market.

In our own prior work, in [13], we have studied the role user interface (UI) design plays for the design of *hidden markets*, and we have described in detail the design and evaluation of a new UI for the P2P backup application. This paper complements that work: we only briefly describe the UI, to the degree necessary to understand our model, and focus on the market design and analysis. This paper is a significantly extended version of a previous workshop contribution [12].

2. THE P2P RESOURCE MARKET

Our system uses a hybrid P2P architecture where all files are transferred directly between peers, but a dedicated server coordinates all operations and maintains meta-data about the location and health of the files. The role of the server in this system is so small that standard load-balancing techniques can be used to avoid scaling bottlenecks.

Each user in the system is simultaneously a *supplier* and a *consumer* of resources. A single peer on the consumer side demanding a service (backup, storage, or retrieval) needs multiple peers on the supplier side offering their resources (space, upload and download bandwidth, and online time). The production process of the server (bundling multiple peers and coordinating them) is essential, turning unreliable storage from individual peers into reliable storage. Note that each peer on the supplier side offers a different resource bundle while each peer on the consumer side gets the same product, i.e., a backup service with the same, high reliability.

Erasure Coding and Replication. One natural concern about P2P backup is that individual users have a much lower availability than dedicated servers. Thus, a P2P system must maintain a higher file redundancy to guarantee the same file availability as server-based systems. Simply storing multiple file copies would be very costly. Fortunately, we can

significantly reduce the replication factor by using *erasure coding* [10]. The erasure code splits up a file into k fragments, and produces $n > k$ new fragments, ensuring that *any* k of the n fragments are enough to reconstruct the file. Using this technique, we can achieve the same high reliability as sever-based systems while keeping replication low. For example, if users are online 12h/day on average, using erasure coding we can achieve a file availability of 99.999% with a replication factor as low as 3.5, compared to simple file replication which would result in a factor of 17.

Note that the process for backing up files always involves four steps. First, the user's files are compressed. Then the compressed files are automatically encrypted with a private key that only the user has access to (via Microsoft LiveID). Then, the encrypted file is erasure coded, and then the individual fragments are distributed over hundreds of peers. Using this process, the security of our P2P backup systems can be made as high as that of any server-based system.

Basic Operations in the Backup System. We consider the following five high-level operations:

1. **Backup**: When a user performs a backup, file fragments are sent from the consumer to the suppliers.
2. **Storage**: Suppliers must persistently store the fragments they receive (until they are asked to erase them).
3. **Retrieval**: When a user retrieves a backup, file fragments are sent from the suppliers to the consumer.
4. **Repair**: When the server determines a backed up file to be unhealthy, the backup is repaired.
5. **Testing**: If necessary, the server initiates test operations to gather new data about a peer's availability.

Table 1: Operations and their Required Resources.

Operation	Resources Required from Suppliers
1. Backup	Download Bandwidth
2. Storage	Space
3. Retrieval	Upload Bandwidth
4. Repair	Download and Upload Bandwidth
5. Testing	Download and Upload Bandwidth

Prices, Trading & Work Allocation. For now, monetary transfers are prohibited and all trades in the market are done using a virtual currency. Each resource has a price at which it can be traded and in each transaction the suppliers are paid for their resources and the consumers are charged for consuming services. Prices are updated regularly according to current aggregate supply and demand, to bring the system into equilibrium over time.

Trading is enabled via a centralized accounting system, where the server has the role of a bank. The server maintains an account balance for each user starting with a balance of zero and allows each user to take on a certain maximal deficit. The purpose of the virtual currency is to allow users to do work at different points in time while maintaining fairness. Users have a steady inflow of money from supplying resources and outflow of money from consuming services, which varies over time. In *steady state*, when users have been online long enough, their income must equal their expenditure. Users cannot earn money when they are offline but must still pay for their backed up files. Thus, their balance continuously decreases during that time. As long as we do not use real money, the maximum deficit that users can take on must be bounded. Ultimately, it is a policy decision what happens when a user hits a pre-defined deficit level. Our system will first notify the user (via email and visually in the application) and present options to remedy the situation (e.g., increase supply). Failing this after a reasonable timeout period (e.g., 4 weeks), the user's backups will be deleted. The server is involved in every operation, coordinating the work done by the suppliers and allocating work to those users with the lowest account balances to drive all accounts (back) to zero over time. This is possible because the users' steady-state income must equal their expenditure. Thus, when users have been online for a sufficient time period, their account balance is always close to zero.

3. THE USER INTERFACE

In this section we only describe the UI to the degree necessary to understand the economic model; see [13] for more details. Figure 1 displays our current implementation of the UI, a "settings window" where users can control their resources. This window has two distinct areas: On the right side, the users can set maximum bounds on the supply they are willing to give up, using the three sliders for space, upload and download bandwidth. Below the sliders the current average online time of the users is displayed. To change this value the users have to leave their computer online for more or fewer hours per day than they are currently doing. On the left side of the window, the users can choose how much online backup space they need. On the bar chart the users can see how much they have already backed up (their current demand) and how much total online backup space they have (the bound on their demand) given the current supply.

To change anything about their settings, the users can either drag the bar chart on the left side up or down, move any of the sliders on the right side, or change how often they are online. Both sides of the window are connected to each other such that a change on either side affects and dynamically updates the values on the other side as well. The semantics of this connection are important: on average, users must be able to pay for the total consumption chosen on the left side with the supply chosen on the right side.

The users have multiple ways of seeing/experiencing current market prices of the resources. They can either move the sliders while observing the bar chart on the left. If the bar chart moves a lot, then the price for that resource is relatively high. If the bar chart only moves a little, the price is currently relatively low. Alternatively, users can look at the help text to the right of the three sliders, where we show the users how much more of each resource they have to give up to get 1 more GB of online backup space. This information is also an indirect encoding of the current resource prices.

Bundle Constraints. We now turn our attention to the combinatorial aspect of the market, the bundle constraints. If a user keeps increasing one slider towards the maximum while the other two sliders are relatively low, at some point the online backup space on the left might stop increasing. For example, if users limit their upload bandwidth to 5 KB/s, then increasing their space supply from 50 GB to 100 GB should not increase their online backup space. We would simply never store 100 GB on these users' hard disks because 5 KB/s would not be enough to have a reasonable retrieval rate for all of these file pieces. Thus, for the sys-

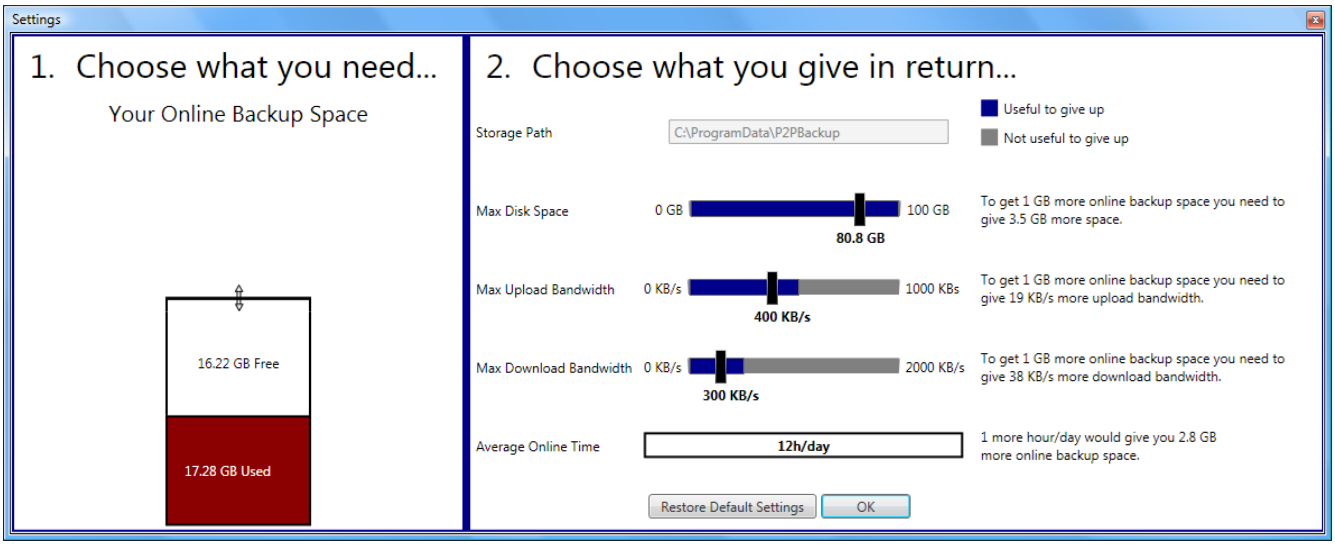

Figure 1: Screenshot of the implemented user interface. On the left side, the users see their current demand and the upper bound on how much they can maximally consume given their supply choice. On the right side, the users can specify bounds on the maximum supply they are willing to give up.

tem to use the whole supply of 100 GB, the users would first have to increase their supply of bandwidth. An analogous argument holds true for other combinations of resources. In the UI, as long as the user moves the sliders within the blue region (where the resources are "useful to give up"), the online backup space changes more or less linearly. Once the user moves the slider out of the blue region into the gray region, the online backup space stops increasing because the user is now supplying more of that resource than is useful.

In our implementation, we use system-wide information regarding the demand for each of the three resources to determine the regions where a user's supply is useful. Obviously, if a user supplies resources in the same ratio as they are being used in the system, then all of the resources are useful. However, we give each user a certain "slack", i.e., we allow the users to specify supply ratios that deviate from the system-wide ratio by a factor $\gamma > 1$ into either direction. Note that the slack factor determines the users' flexibility regarding their resource supply. This flexibility comes from the fact that the server can navigate certain kinds of work to certain users. For example, repair traffic consumes a lot of bandwidth but no space; cold backups require a lot of space but only little bandwidth; hot backups require a lot of bandwidth but little space. The server chooses the slack factor depending on how much freedom the system has in allocating different kinds of work to different kinds of users.

4. THE ECONOMIC MODEL

In this section we introduce a formal economic model to allow for a theoretical analysis of the properties of the real P2P market system. In the P2P economy, there are I users who are simultaneously suppliers and consumers. The set of commodities traded on the market is $L = \{S, U, D, A, B, \Sigma, R\}$. The first four commodities are space (S), upload bandwidth (U), download bandwidth (D), and availability (A), which are the resources that users supply. The last three commodities are backup service (B), storage service (Σ), and retrieval service (R), which are the services that users consume. By slightly abusing notation, we sometimes use S, U, D, etc. as subscripts and sometimes they denote the resource space, e.g., for a particular amount of upload bandwidth u we require that $u \in U$. Each user i has a fixed endowment of the supply resources denoted $w_i = (w_{iS}, w_{iU}, w_{iD}, w_{iA}) \in S \times U \times D \times [0, 1]$. We analyze the equilibrium in a single snapshot of the market when all users adjust their bounds on supply to reach their desired demand vectors.

The next aspect of the model is driven by our UI. Via the sliders, the user selects upper bounds for the supply vector, which we denote $X_i = (X_{iS}, X_{iU}, X_{iD}, X_{iA})$. In return for the supply X_i, the user interface shows the user the maximum demand of services, denoted $Y_i = (Y_{iB}, Y_{i\Sigma}, Y_{iR})$. In Figure 1 you can see that this user has currently chosen $X_{iS} = 80.8GB$, $X_{iU} = 400KB/s, X_{iU} = 300KB/s$ and $X_{iA} = 0.5$ as the maximum supply vector.

At any point in time, a certain set of resources from the user are being used, always less than X_i, and a certain set of services is being demanded. We denote user i's current supply as $x_i = (x_{iS}, x_{iU}, x_{iD}, x_{iA})$, and analogously user i's current demand for services as $y_i = (y_{iB}, y_{i\Sigma}, y_{iR})$. The user does not choose x_i and y_i directly via the UI. Instead, the server chooses x_i (obeying the bound X_i) such that user i can afford the current demand y_i which the user simply chooses by backing up files or retrieving them. Note that the UI displays the user's consumption vector in an aggregated way; i.e., instead of listing the services backup, storage, and retrieval separately, we simply display the currently used online backup space ($= 17.28GB$ in Figure 1) and the maximum online backup space that user could consume ($= 33.5GB$ in Figure 1).

In practice, users have a certain cost for opening the settings window and adjusting the settings. Instead of modeling this cost factor directly, we assume that when users open their settings window, they are planning ahead for the whole time period until they plan to open the settings window the next time. While a user might currently consume y_i, he plans for consuming up to Y_i the next time he opens the settings window. He then selects the supply vector X_i that he is willing to give up to get this Y_i. The user cares about how

large the bounds on his supply are, because he has negative utility for giving up his resources. To make this more formal, we let $K_i = w_i - X_i$ with $K_i \in S \times U \times D \times A$, denote the vector of resources that the user keeps, i.e., his endowment minus the supply he gives up. We can specify the user's preference relation over all the resources he keeps, and the services he consumes: $\succeq_i (K_{iS}, K_{iU}, K_{iD}, K_{iA}, Y_{iB}, Y_{i\Sigma}, Y_{iR})$. We make the following assumption (cf. [11], chapters 1-3):

Assumption 1. Each user's preferences are (i) complete, (ii) transitive, (iii) continuous, (iv) strictly convex, and (v) monotone.

Note that strict convexity requires marginal rates of substitution between two goods, i.e., the user needs to get more and more of one good when we take away 1 unit of another good. This is a reasonable assumption for the user's resources and services because it represents a preference for diversification. Monotonicity means that all commodities are "goods", i.e., if we give users more of any of the commodities, they are at least as well off as before. Given complete, transitive, and continuous preferences, there exists a utility function $u_i(K_i, Y_i) = u_i(K_{iS}, K_{iU}, K_{iD}, K_{iA}, Y_{iB}, Y_{i\Sigma}, Y_{iR})$ that represents the preference relation and this utility function is continuous (cf. [11], p.47).

4.1 Prices and Flow Constraints

The system can avoid non-linear prices and support an equilibrium with linear prices as long as the user specifies a supply vector within the slack region that the system allows for each user's supply. The only resource that is not subject to these slack regions, or limits, is *availability*: as long as the user's availability is larger than zero, the other resources can be used. To simplify the pricing model, we introduce three new *composite resources* $\overline{S}, \overline{U}$, and $\overline{D}$, incorporating the user's availability into the other resources in the following way:

- $\overline{X}_{i\overline{U}} \in \overline{U} = X_{iU} \cdot X_{iA} \cdot 24 \cdot 60 \cdot 60.$
- $\overline{X}_{i\overline{D}} \in \overline{D} = X_{iD} \cdot X_{iA} \cdot 24 \cdot 60 \cdot 60.$
- $\overline{X}_{i\overline{S}} \in \overline{S} = \overline{\varphi}(X_{iS}, X_{iA}) \approx X_{iS} \cdot X_{iA} \cdot$ overhead factor

Note that this notation denotes *composite* and *not* vector quantities. The definitions for the composite resources upload and download bandwidth are straightforward: we multiply the bound on bandwidth the user supplies (e.g., 300 KB/S) with the user availability $\in [0, 1]$ and then multiply it with 24 hours, 60 minutes and 60 seconds, to calculate how many KBs we can actually send to this user per day. The definition of $\overline{X}_{i\overline{S}}$ is a little more intricate because the user's availability does not enter linearly into the calculation. However, it enters monotonically, i.e., more availability is always better. Here, it suffices to know that the server can compute this function $\overline{\varphi}$ and convert a user's space and availability supply into the new composite resource; further details are beyond this paper.

We can now define user i's supply vector for the three composite resources: $\overline{X_i} = (\overline{X_{iS}}, \overline{X_{i\overline{U}}}, \overline{X_{i\overline{D}}})$. The advantage of using these composite resources is that now, the supply from different users with different availabilities is comparable. For example, 1 unit of $\overline{S}$ from agent i with availability 0.5 is now equivalent to 1 unit of $\overline{S}$ from agent j with availability 0.9. Obviously, internally agent i has to give much more space to make up for his lower availability, but in terms of bookkeeping, we can now operate directly with composites. We define the aggregate supply vector for the composite resources as $\overline{X} = \sum_i \overline{X_i}$, and analogously for Y, $\overline{x}$ and y. We make the following well-known observation (cf. [11], chapter 3) that will be useful later:

Observation 1. The supply and demand functions $\overline{X_i}$ and Y_i are homogeneous of degree zero. This implies that the aggregate supply and demand functions, i.e., $\overline{X}$ and Y are also homogeneous of degree zero.

Now we get to the pricing aspect of the system. We use $p = (p_{\overline{S}}, p_{\overline{U}}, p_{\overline{D}})$ for the prices for supplied composite resources, and $q = (q_B, q_\Sigma, q_R)$ for the demanded services. We require that in steady state, users can pay for their consumption with their supply. We can express this *flow constraint* formally:

$$\overline{X}_i \cdot p = Y_i \cdot q. \tag{1}$$

At the same time, the server allocates enough work to user i such that the user's current supply $\overline{x}_i$ is enough to pay for the demand y_i, which leads to a second flow constraint:

$$\overline{x}_i \cdot p = y_i \cdot q. \tag{2}$$

4.2 Production Functions

We have already mentioned the important role of the server in our market, i.e., that of combining resources from different suppliers into a valuable bundle. Formally, the server is the only producer in our market.[3] For each service, we have a production function that defines how many input resources are needed to produce one unit of that service:

- Backup: $f_B : \overline{S} \times \overline{U} \times \overline{D} \to B$
- Storage: $f_\Sigma : \overline{S} \times \overline{U} \times \overline{D} \to \Sigma$
- Retrieval: $f_R : \overline{S} \times \overline{U} \times \overline{D} \to R$

Note that these production functions are defined via the implementation of our system, i.e., the particular production technology that we implemented. For example, they are defined via the particular erasure coding algorithm that is being used, by the frequency of repair operations, etc. Thus, we can now specify a series of properties that these production functions guarantee due to our implementation:

System Property 1. Production functions are fixed and the same for all users.

System Property 2. The production functions all exhibit constant returns to scale (they are homogeneous of degree 1), i.e., $\forall l \in \{B, \Sigma, R\} : f_l(k \cdot a, k \cdot b, k \cdot c) = k \cdot f_l(a, b, c) \forall k \in \Re$.

System Property 3. Each production function is bijective, and thus we can take the inverses:

- $f_B^{-1} : B \to \overline{S} \times \overline{U} \times \overline{D}$
- $f_\Sigma^{-1} : \Sigma \to \overline{S} \times \overline{U} \times \overline{D}$
- $f_R^{-1} : R \to \overline{S} \times \overline{U} \times \overline{D}$

Given the inverse functions for the individual services backup, storage, and retrieval, we can define an inverse function for a three-dimensional service vector $(b, \sigma, r) \in B \times \Sigma \times R$:

$$f^{-1}(b, \sigma, r) = f_B^{-1}(b) + f_\Sigma^{-1}(\sigma) + f_R^{-1}(r) \tag{3}$$

[3]Note that this is what allows us to define an exchange economy despite the fact that production is happening in the market. For more details see [11], pp. 583-584.

Property 1 holds because of the way we have defined the composite resources, with any differences between the agents' availabilities already considered. Property 2 (CRTS) holds approximately for file sizes above a certain threshold (approx. 1MB) due to the properties of the erasure coding algorithm.[4] Property 3, the bijectivity of production, holds, because for each service unit, there is only one way to produce it. For example, to backup one file fragment, the erasure coding algorithm tells us exactly how many supplier fragments we need, and the server tells us how much repair and testing traffic we can expect on average per fragment.

The following system's property comes from the UI design and is motivated by keeping the money flow in the system constant. This property is only possible because the server is the only producer in our system and production functions are fixed (Property 1), exhibit constant returns to scale (Property 2), and are bijective (Property 3):

System Property 4. We charge the consumers exactly the amount we pay the suppliers, i.e., for demand vector y_i, we charge user i exactly:

$$y_i \cdot q = f^{-1}(y) \cdot p.$$

Using Property 4, we can now re-write the flow constraints for agent i as:

$$\overline{X_i} \cdot p = f^{-1}(Y_i) \cdot p \quad \text{and} \quad \overline{x_i} \cdot p = f^{-1}(y_i) \cdot p$$

Thus, from now on, we can omit the price vector q for demanded services and only need to consider price vector p.[5] Effectively, we can treat the whole market as an exchange economy for the composite resources $\overline{S}, \overline{U}$, and $\overline{D}$, assuming users only engage in exchange of supplied resources because everyone has access to the same production technology (cf. [11], pp. 582-584).

Remember that the UI automatically calculates and adjusts the maximum demand vector Y_i for user i based on the user's supply bound $\overline{X}_i$. In practice, the maximum income is divided by the current average income of the user, and the resulting factor is multiplied with the user's current demand, giving us the maximum demand the user can afford:

System Property 5. The system uses a linear demand prediction model for the calculation of a user's maximum demand Y_i:

$$Y_i = \frac{\overline{X}_i \cdot p}{\overline{x}_i \cdot p} \cdot y_i = \lambda_i \cdot y_i$$

To facilitate the equilibrium analysis in the next section, we make the following simplifying assumption:

Assumption 2. We assume that with a large number of users, a linear demand prediction is also correct for the aggregate demand vectors, i.e.:

$$\exists \lambda : Y = \lambda \cdot y$$

[4]Very small files are an exception and need special treatment in the implementation, because they are more expensive to be produced (again due to the erasure coding). We take care of this in the implementation by charging users more when they are backing up small files (essentially we have two sets of prices, one for normal files and one for small files).

[5]Going forward, please remember that multiplications with p are always dot products, and thus p showing up on the left and the right side of an equation does not cancel out.

This assumption is justified because in practice, the system will have thousands or millions of users. Let n denote the number of users in the economy, let $Y^n = \sum_{i=1}^{n} Y_i$, $y^n = \sum_{i=1}^{n} y_i$, and let $\mu(\lambda_i)$ denote the mean of the distribution of the λ_i's. Given that the λ_i's are independent from the y_i's, it follows from the strong law of large numbers, that if the number of users n is large enough, then Y^n is linearly predictable by $\mu(\lambda_i) \cdot y^n$ along each dimension to any additive error. More specifically, for any ε and $\delta \geq 0$, for large enough n:

$$Pr[||Y^n - \mu(\lambda_i) \cdot y^n|| \leq \varepsilon] \geq 1 - \delta.$$

5. EQUILIBRIUM ANALYSIS

A real-world instance of the P2P backup application would have thousands if not millions of users. Thus, the underlying market would be large enough so that no individual agent had a significant effect on market prices. Consequently, users can be modeled as price-taking agents and a general equilibrium model is suitable to analyze this market.

5.1 The Buffer Equilibrium

A standard equilibrium concept in general equilibrium theory is the Walrasian equilibrium which requires that demand equals supply such that the market clears. Certainly we want to have enough supply to satisfy current demand, i.e., we want that:

$$\overline{x} = f^{-1}(y).$$

But remember that users are not constantly adjusting $\overline{x}_i$. Instead, they choose maximum bounds on their supply $\overline{X}_i$ via the UI. But given that the maximum supply $\overline{X}_i$ is generally larger than $\overline{x}_i$, it is not a very strong requirement to have $\overline{x} = f^{-1}(y)$. In particular, we do not only want to balance the market *now*, but we want to guarantee that the backup system can also satisfy demand Y in the future, which implies that we must always have some excess supply of all resources. Ideally, we want to maximize the buffer between the current usage of resources, i.e., $f^{-1}(y)$ and the maximum supply of resources, i.e., $\overline{X}$. We will use this "size of the buffer" repeatedly and thus define it more formally:

Definition 1. (Size of the Supply-Side Buffer) The size of the supply-side buffer is the smallest ratio, over all resources, of maximum supply to current demand:

$$\lambda = \min_{l \in \{\overline{S}, \overline{U}, \overline{D}\}} \frac{\overline{X}_l}{f_l^{-1}(y)} \tag{4}$$

The reason for having a supply side buffer is that we want to be safe, i.e., we want to be sure that we can satisfy new incoming requests. More specifically, we want to make sure that as demand increases from its current state y to the maximum state we allow the users Y, we will always have enough supply to satisfy this demand. More formally:

Definition 2. (Safety Property) The safety property of the system is that we always have enough supply to satisfy increasing demand, i.e.:

$$\forall y \leq Y : \overline{X} \geq f^{-1}(y) \tag{5}$$

Using the bijectivity and the CRTS properties of the production functions, it is easy to show the following lemma:

LEMMA 1. *If $\overline{X} = f^{-1}(Y)$, then the safety property is satisfied.*

In words, when the bound on aggregate supply of all resources equals the amount of resources needed to produce the projected service vector Y, then we can guarantee the safety property. When we have reached this state of the system, we say we have reached the *buffer equilibrium*:

Definition 3. (Buffer Equilibrium) A Buffer equilibrium is a price vector $p = (p_{\overline{S}}, p_{\overline{U}}, p_{\overline{D}})$ an aggregate supply vector $\overline{X}(p)$ and an aggregate demand vector $Y(p)$ such that:

$$\overline{X}(p) = f^{-1}(Y(p))$$

i.e., it is a Walrasian equilibrium defined on the supply and demand bounds chosen by the users.

We call this equilibrium the "buffer equilibrium" because the extent to which $\overline{X}$ is above $f^{-1}(y)$, i.e., the size of the buffer, determines the "level of safety" in the system.

5.2 Equilibrium Existence

In this section, we will prove that a buffer equilibrium exists under some reasonable assumptions. To do so, we first introduce some new notation and prove two Lemmas before we get to the actual theorem. We let $\overline{L} = \{\overline{S}, \overline{U}, \overline{D}\}$ and we use l to index a particular composite resources. We define the vector-valued function $Z(p)$ to measure the relative buffer for each individual resource in the following way:

$$Z_l(p) = \Big(\frac{\sum_l \frac{\overline{X}_l(p)}{f_l^{-1}(y(p))}}{|\overline{L}|}\Big) - \frac{\overline{X}_l(p)}{f_l^{-1}(y(p))} \tag{6}$$

In words, the first term represents the average supply to demand ratio, in our case averaged over the three goods storage space, upload and download bandwidth. The second term represents the supply to demand ratio of the particular good l. Thus, $Z_l(p)$ represents how far the "buffer" between supply and demand for good l is away from the average buffer. If Z_l is negative, then the buffer between supply and demand for good l is relatively high and should be decreased; if Z_l is positive, then the buffer between supply and demand for good l is relative low and should be increased. If the buffer is the same for all goods, we have reached the equilibrium. Thus, we can prove the following Lemma:

LEMMA 2. *If $Z(p) = 0$, then the market has reached a buffer equilibrium and p is the equilibrium price vector.*

PROOF. If $Z(p) = 0$ then:

$$\forall l : \Big(\frac{\sum_l \frac{\overline{X}_l(p)}{f_l^{-1}(y(p))}}{|\overline{L}|}\Big) = \frac{\overline{X}_l(p)}{f_l^{-1}(y(p))}$$
$$\Rightarrow \exists \lambda > 1 \text{ s.t. } \forall l : \overline{X}_l(p) = \lambda \cdot f_l^{-1}(y(p))$$

Now, due to Assumption 2 we know that $\exists \delta : Y = \delta \cdot y$. Thus:

$$\Rightarrow \forall l : \overline{X}_l(p) = \lambda \cdot f_l^{-1}(\frac{1}{\delta} Y(p)) \tag{7}$$

$$\Rightarrow \forall l : \overline{X}_l(p) = \lambda \cdot \frac{1}{\delta} \cdot f_l^{-1}(Y(p)) \tag{8}$$

$$\Rightarrow \forall l : \overline{X}_l(p) = \lambda^* \cdot f_l^{-1}(Y(p)) \quad \text{for } \lambda^* = \lambda \cdot \frac{1}{\delta} \tag{9}$$

$$\Rightarrow \overline{X}(p) = \lambda^* \cdot f^{-1}(Y(p)) \tag{10}$$

But from the flow constraints (Eqn. 4) we also know that:

$$\overline{X}(p) \cdot p = f^{-1}(Y(p)) \cdot p \tag{11}$$

Equations (10) and (11) can only both be true if $\lambda^* = 1$. Thus, it follows that:

$$\overline{X}(p) = f^{-1}(Y(p))$$

which is the definition of the buffer equilibrium. □

Next we show that $Z(\cdot)$ has a series of nice properties:

LEMMA 3. *The function $Z(\cdot)$ has the following properties:*

(i) *$Z(\cdot)$ is continuous.*

(ii) *$Z(\cdot)$ is homogeneous of degree zero.*

(iii) *$\forall p : \sum_l Z_l(p) = 0$.*

(iv) *If $p^n \to p$, where $p \neq 0$, $p_l > 0$ and $p_k = 0$ for some $k \neq l$, then for n sufficiently large:*

$$Z_k(p^n) = \max\{Z_{\overline{S}}(p^n), Z_{\overline{U}}(p^n), Z_{\overline{D}}(p^n)\}.$$

PROOF. Property (i), the continuity of $Z(\cdot)$ follows directly from the continuity of the user preferences (which is why $\overline{X}(p)$ and $y(p)$ are continuous) and the continuity of the inverse production functions. Property (ii), the homogeneity of degree zero follows because $\overline{X}(p)$ and $y(p)$ are homogeneous of degree zero. Property (iii), follows directly from the definition of $Z(\cdot)$:

$$\sum_l Z_l(p) = 3 \cdot \Big(\frac{\sum_l \frac{\overline{X}_l(p)}{f_l^{-1}(y(p))}}{|\overline{L}|}\Big) - \sum_{l \in \{\overline{S}, \overline{U}, \overline{D}\}} \frac{\overline{X}(p)}{f_l^{-1}(y(p))} = 0$$

Finally, property (iv): as the price of resource $k \in \{\overline{S}, \overline{U}, \overline{D}\}$ goes towards zero, due to users' strongly monotone preferences for supply resources, they will supply less and less of that resource, and supply more of the other resources instead, at least of resource l whose price is bounded away from zero. However, because of the bundle constraints, the users cannot reduce their supply of resource k towards zero. Let $\gamma > 1$ denote the slack factor we allow users when setting their preferences. The relevant constraints, lower-bounding the supply for resource k, are:

$$\forall l \in \overline{L} \setminus \{k\} : \overline{X}_{ik} \geq \frac{1}{\lambda} \cdot \frac{f_k^{-1}(Y)}{f_l^{-1}(Y)} \cdot \overline{X}_{il}$$

As $p^n \to p$ with $p_k = 0$, for n large enough, p^n will be sufficiently close to zero, such that each user i chooses to supply the minimal amount of resource k that is possible. Thus, at least with respect to one of the other resources, the slack constraint will be binding, i.e.,:

$$\overline{X}_{ik} = \max\Big\{\frac{1}{\lambda} \cdot \frac{f_k^{-1}(Y)}{f_l^{-1}(Y)} \cdot \overline{X}_{il}, \frac{1}{\lambda} \cdot \frac{f_k^{-1}(Y)}{f_m^{-1}(Y)} \cdot \overline{X}_{im}\Big\}$$

This does not say that the constraint will be binding for the same resource l or m for every user. However, for every user, one of the constraints will be biding and thus, every user will contribute least to the supply side buffer for resource k. Consequently, the total supply side buffer for good k will be minimal (i.e., $Z_k(p^n)$ will be maximal), which implies that $Z_k(p^n) = \max\{Z_{\overline{S}}(p^n), Z_{\overline{U}}(p^n), Z_{\overline{D}}(p^n)\}$. □

THEOREM 1. *A buffer equilibrium exists in the P2P exchange economy, given that users' preferences are continuous and strictly convex, monotone w.r.t. service products as well as strongly monotone w.r.t. to supply resources.*

PROOF. We have shown in Lemma 2 that once we have found a price vector p such that $Z(p) = 0$, we have reached a buffer equilibrium. Furthermore, in Lemma 3 we have shown four properties of $Z(\cdot)$. Equipped with these two results, the remainder of our proof follows using techniques from the standard equilibrium existence proof for the Walrasian Equilibrium (see [11] page 586). We omit the details here due to space constraints, but we want to briefly point out where changes in the proof are necessary. First, note that we are not working with the excess demand function of this economy and instead use the function $Z(\cdot)$ that measures the relative buffer size for each resource. Thus, in step 1 of the proof in [11], we cannot use Walras' law and instead use property (iii) of Lemma 3. Second, in step 4 of the proof in [11], when proving upper hemicontinuity of the fixed-point correspondence, we cannot use the result that the excess demand for one resources goes to infinity when its price goes towards zero. Instead, we use property (iv) of Lemma 3. □

One might wonder what prevented the direct applicability of standard theorems regarding equilibrium existence (e.g., [11] page 585). It turns out that the standard results were not directly applicable for three reasons. Most importantly, we do not assume that users have strongly monotone preferences w.r.t. service products. The consequence of this is that as the price of one resources goes towards zero, it is not necessarily the case that the demand for service products produced from that resource go towards infinity. Second, we do not have a pure exchange economy and have to take the production functions into account. Those exhibit constant returns to scale which implies that production sets are neither strictly convex nor bounded above, which complicates the analysis significantly (cf. [11] page 583). Third, each user's supply is subject to the bundle constraints, i.e., a user's supply cannot drop below or go above certain limits. Due to these three factors, we needed slightly different machinery to prove equilibrium existence in our economy.

5.3 Equilibrium Uniqueness

Without any further restrictions on the user's preferences, we cannot say anything about the uniqueness of the buffer equilibrium (cf. Sonnenschein-Mantel-Debreu Theorem, [11], pp. 598-606), because the substitution effect and the wealth effect could either go in the same direction or in opposite directions.The gross substitutes assumption resolves this problem, by assuming that in gross terms, taking substitution and wealth effect into account, the resources are substitutes. We assume that this is the case for the supply resources:

Assumption 3. (Supply Resources are Gross Substitutes) We assume that the aggregate supply function $\overline{X}(p)$ satisfies the gross substitutes condition [1], i.e., whenever p' and p are such that, for some k, $p'_k > p_k$ and $p'_l = p_l$ for $l \neq k$, we have $\overline{X}_l(p') < \overline{X}_l(p)$ for $l \neq k$.

The standard equilibrium uniqueness proof for Walrasian equilibria relies on the assumption that the aggregate excess demand function satisfies the gross substitutes condition for *all* commodities. However, we only want to make that assumption w.r.t. supply resources where this seems very reasonable because as the price for one resource decreases, this means that the relative price for another resource increases, and thus users would be happy to supply more of the more costly resources now. However, for the demanded services, the gross substitutes assumption is most certainly violated. For example, if the price for storage would increase, it is not reasonable to assume that now users would store fewer files online, but instead consume more backup and retrieval operations. Thus, we cannot make the gross substitutes assumption for all commodities. Instead, we will make the following assumption w.r.t consumed services:

Assumption 4. (Services are Perfect Complements) We assume that the aggregate demand function $Y(p)$ satisfies the perfect complements condition.

A consequence of the perfect complements condition is that price changes affect all dimensions of the aggregate demand vector equally. For an individual user, the Leontief utility function would induce the perfect complements property. However, note that we do not require individual users to have demand functions that satisfy the perfect complements condition. It is a much weaker assumption, and much more reasonable due to the law of large numbers, to assume that the *aggregate* demand function satisfies it.

THEOREM 2. *The buffer equilibrium is unique (up to normalization), given that the aggregate supply function satisfies the gross substitutes property (Assumption 3), and that the aggregate demand function satisfies the perfect complements property (Assumption 4).*

PROOF. The fact that we have different assumptions on the supply and demand side of our economy complicates the uniqueness proof. When prices go up for good l, it is not a priori clear what happens to the buffer equilibrium. To get a better handle on this, we first separate the supply and demand aspects by introducing yet another alternative description of the buffer equilibrium:

$$\overline{X} = f^{-1}(Y) \tag{12}$$

$$\Leftrightarrow \left(\overline{X}_{\overline{S}}, \overline{X}_{\overline{U}}, \overline{X}_{\overline{D}}\right) = \left(f_{\overline{S}}^{-1}(Y), f_{\overline{U}}^{-1}(Y), f_{\overline{D}}^{-1}(Y)\right) \tag{13}$$

$$\Leftrightarrow \left(1, \frac{\overline{X}_{\overline{U}}}{\overline{X}_{\overline{S}}}, \frac{\overline{X}_{\overline{D}}}{\overline{X}_{\overline{S}}}\right) = \left(1, \frac{f_{\overline{U}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}, \frac{f_{\overline{D}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}\right) \tag{14}$$

$$\Leftrightarrow \left(\frac{\overline{X}_{\overline{U}}}{\overline{X}_{\overline{S}}}, \frac{\overline{X}_{\overline{D}}}{\overline{X}_{\overline{S}}}\right) - \left(\frac{f_{\overline{U}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}, \frac{f_{\overline{D}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}\right) = 0 \tag{15}$$

We define a new vector-valued function $g(p) = \left(g_{\overline{U}}(p), g_{\overline{D}}(p)\right)$:

$$g_{\overline{U}}(p) = \left(\frac{\overline{X}_{\overline{U}}}{\overline{X}_{\overline{S}}} - \frac{f_{\overline{U}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}\right) \quad \text{and} \quad g_{\overline{D}}(p) = \left(\frac{\overline{X}_{\overline{D}}}{\overline{X}_{\overline{S}}} - \frac{f_{\overline{D}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}\right),$$

which naturally leads to a new equilibrium definition:

Definition 4. (Buffer Equilibrium [1. Alternative]) A buffer equilibrium is a price vector p and $g(p)$ such that

$$g(p) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Thus, we have simplified the problem of finding equilibrium prices to finding the root of the function $g(p)$. Showing uniqueness of the buffer equilibrium is now equivalent to

showing that $g(p) = 0$ has at most one (normalized) solution. Now, let's assume that $g(p) = 0$, i.e., p is an equilibrium price vector. We show that for any p', $g(p') \neq 0$ unless p and p' are collinear, i.e., unless $p' = \lambda p$ for some $\lambda > 0$. Note that because $\overline{X}(p)$ and $Y(p)$ are homogeneous of degree zero, $g(\cdot)$ is also homogeneous of degree zero. Thus, we can assume that $p' \geq p$ and $p_l = p'_l$ for some l. We now alter the price vector p' to obtain p in two steps, lowering (or keeping unaltered) the price of resources $k \neq l$ one at a time. Because of Assumption 4 (the aggregate demand function satisfies the perfect complements condition), a price change affects all dimensions of the demand function equally, i.e., $\exists \mu \in \mathbb{R} : Y(p) = \mu \cdot Y(p')$. Because the production function is bijective and exhibits constant returns to scale, this implies that $f^{-1}(Y(p)) = \mu \cdot f^{-1}(Y(p'))$. Thus, $\frac{f_{\overline{U}}^{-1}(Y(p))}{f_{\overline{S}}^{-1}(Y(p))} = \frac{f_{\overline{U}}^{-1}(Y(p'))}{f_{\overline{S}}^{-1}(Y(p'))}$, i.e., changes in the demand function $Y(\cdot)$ due to price changes do not affect $g(\cdot)$. Thus, we only have to pay attention to changes in the supply function $\overline{X}$. Here, we need to differentiate the following 3 cases:.

Case 1: l =storage. By gross substitution (see Assumption 3), the supply of good $\overline{S}$ cannot decrease in any step, and, because $p \neq p'$, it will actually increase in at least one step. In turn, the supply of $\overline{U}$ and $\overline{D}$ will stay the same or decrease because of homogeneity of degree zero. Thus, the first term in the $g(\cdot)$ functions will decrease, while the second term stays constants, and thus, $g(p) < g(p')$.

Case 2: l =upload bandwidth. By gross substitution, the supply of good $\overline{U}$ cannot decrease in any step, and, because $p \neq p'$, it will actually increase in at least one step. The supply of $\overline{S}$ and $\overline{D}$ on the other hand will stay the same or decrease. Thus, the first term in $g_{\overline{U}}(\cdot)$ will increase, while the second term stays constants. Thus, $g_{\overline{U}}(p) > g_{\overline{U}}(p')$ (note, we do not even need to consider $g_{\overline{D}}(p)$ in this case).

Case 3: l =download bandwidth. By gross substitution, the supply of good $\overline{D}$ cannot decrease in any step, and, because $p \neq p'$, it will actually increase in at least one step. The supply of $\overline{S}$ and $\overline{U}$ on the other hand will stay the same or decrease. Thus, the first term in $g_{\overline{D}}(\cdot)$ will increase, while the second term stays constants. Thus, $g_{\overline{D}}(p) > g_{\overline{D}}(p')$ (again, we do not even need to consider $g_{\overline{U}}(p)$ is this case).

In summary, in all three cases we established that $g(p) \neq g(p')$ which concludes the equilibrium uniqueness proof. □

5.4 Limited Controllability of the Buffer Size

So far we have shown under what conditions the buffer equilibrium exists and when it is unique. We know from Lemma 1 that when the system is in the buffer equilibrium, then the safety property is guaranteed, i.e., we always have enough supply to satisfy demand as it increases from y towards Y. But what happens if the system is out of equilibrium? Note that in practice, users do not permanently adjust their settings, and thus price changes will only affect supply and demand with a significant delay. Consequently, it would be desirable to have a large enough buffer between current demand and maximum supply, such that even if the system is out of equilibrium, we can satisfy new incoming demand. For example, it seems like desirable goal to have at least twice as much supply as current demand, i.e., $\overline{X} \geq 2 \cdot f^{-1}(y)$. Unfortunately, the uniqueness of the buffer equilibrium has an immediate consequence regarding the limited controllability of the buffer equilibrium:

COROLLARY 1. *(Limited Controllability of the Market) Given Property 4, and Assumptions 3 and 4, the market operator cannot influence the size of the buffer in the buffer equilibrium.*

Given this limited controllability, it is natural to ask what buffer size to expect in equilibrium. It turns out that, in equilibrium, the supply side buffer is uniquely determined via the individual demand side buffers of all users.

PROPOSITION 1. *In the buffer equilibrium, given Assumption 2, the size of the supply buffer equals the size of the demand buffer.*

PROOF.

$$\overline{X} = f^{-1}(Y) \tag{16}$$

$$\Leftrightarrow \overline{X} = f^{-1}(\lambda \cdot y) \tag{17}$$

$$\Leftrightarrow \overline{X} = \lambda \cdot f^{-1}(y) \tag{18}$$

□

In words, the size of the buffer depends on how forward-looking the agents are. If on average the users give themselves a 25% buffer on the demand side (e.g., a user has currently backed up 20GB and sets the sliders in such a position that his/her maximum online backup space is 25GB), then we would also have a 25% buffer on the supply side, i.e., $\overline{X} = 1.25 \cdot f^{-1}(y)$.

Now we turn to the question why the market operator cannot influence the size of the supply side buffer, i.e., which system properties or which assumptions we made in our market economy are the limiting ones. Remember that the limited controllability of the buffer equilibrium was a corollary of the uniqueness property, which relied on two assumption, namely gross substitutability of supplied resources, and that services are perfect complements. It turns out, however, that the limited controllability remains even without those assumptions, strengthening the result from Corollary 1:

PROPOSITION 2. *Given system property 4, if each individual user i has a limited planning horizon in that he chooses not to give himself more than a demand side buffer of λ_i, then there exists a Λ such that the market operator cannot achieve a buffer equilibrium with buffer size Λ.*

PROOF. For the proof we construct a simple counterexample. We allow price changes to affect $\overline{x}, \overline{X}, y$ and Y and in particular we do not make the gross substitutability assumption or the perfect complements assumption. We choose a Λ such that $\forall i : \Lambda > \lambda_i$. And we let $\lambda_i^* = \max_i \lambda_i$. Now:

$$\forall i : Y_i = \lambda_i \cdot y_i \tag{19}$$

$$\Rightarrow Y = \sum_i \lambda_i \cdot y_i \tag{20}$$

$$\Rightarrow Y \leq \sum_i \lambda_i^* \cdot y_i \tag{21}$$

$$\Rightarrow Y \leq \lambda_i^* \sum_i \cdot y_i \tag{22}$$

$$\Rightarrow Y \leq \lambda_i^* y \tag{23}$$

$$\Rightarrow f^{-1}(y) \leq \lambda_i^* f^{-1}(y) \tag{24}$$

$$\Rightarrow \overline{X} \leq \lambda_i^* f^{-1}(y) \tag{25}$$

Thus, the buffer between supply and demand would be less or equal to λ_i^* which by assumption was strictly less than the buffer Λ that the market operator desired. □

5.5 Decoupling Supply and Demand Prices

We have seen in the previous section that the limited controllability of the buffer equilibrium does not hinge on the gross substitutes or perfect complements assumptions. In this section we show that the crux of the matter is System Property 4, i.e., the coupling of supply and demand prices:

$$\overline{X_i} \cdot p = Y_i \cdot q \tag{26}$$

$$\Leftrightarrow \overline{X_i} \cdot p = f^{-1}(Y_i) \cdot p \tag{27}$$

So far, we have charged each consumer exactly as much as we pay the suppliers for the corresponding resources. If we decouple supply and demand prices we gain additional freedom in pricing commodities. In particular, if we wanted to increase the supply side buffer, then we could make some services more expensive, i.e., we could increase the price on the services beyond the true costs for the resources necessary to produce them. To avoid extracting money out of the system over time, in an actual transaction, we would still charge the consumers according to the true costs of the resources. The inflated prices would only be used in the UI to induce users to increase their supply so that the market operator could achieve the desired size of the supply buffer.

PROPOSITION 3. *If we decouple supply and service prices, then the market operator can adjust prices such as to achieve any desired buffer size* $\Lambda > 1$.

PROOF. We still let p denote the price used to calculate the payments for the suppliers. But consumers now have to pay price $p' = \Lambda \cdot p$ for the resources that are necessary to produce their services. Thus, the new flow constraint is:

$$\overline{X} \cdot p = f^{-1}(Y) \cdot \Lambda \cdot p \tag{28}$$

Note that showing consumers a price of $\Lambda \cdot p$ instead of p has the same effect as if the amount of resources necessary to produce the corresponding services had increased by factor Λ. Now, assume the market operator updates price vector p, as before, until $Z(p) = 0$. In the proof for Lemma 2 we have shown that this implies $\overline{X}(p) = \lambda^* \cdot f^{-1}(Y(p))$. If we plug this into the new flow constraint, we get:

$$\begin{aligned} \lambda^* \cdot f^{-1}(Y(p)) \cdot p &= f^{-1}(Y) \cdot \Lambda \cdot p \\ \Rightarrow \lambda^* &= \Lambda \\ \Rightarrow \frac{\overline{X}(p)}{f^{-1}(Y(p))} &= \Lambda \\ \Rightarrow \frac{\overline{X}(p)}{f^{-1}(y(p))} &\geq \Lambda \end{aligned}$$

which shows that supply side buffer Λ can be achieved. □

At first sight this result might seem like the perfect solution regarding the previously limited controllability of the buffer equilibrium. But unfortunately, it is not. There is a good reason for charging the consumers as much as we pay the suppliers, namely such that in the UI we can display to the users the true costs of the services they are consuming. Effectively, the UI shows the users some kind of *contract*: "if you want to consume these services, then you need to supply this many resources to pay for them." Note that if we would artificially inflate prices, we would display an incorrect contract, and in some sense "lie" to our users. Some users might actually figure this out over time an then try to exploit it. Furthermore, this technique could actually decrease the efficiency of the system. Some users might consume less services than before because they cannot afford to give up as much supply as the UI says they would need to. In the worst case, users might decide to completely leave the system when the perceived costs for using it seem to high. Thus, it remains to be studied how new user interfaces can be designed that give the market operator more control over the supply side buffer, while avoiding the problems we mentioned.

6. THE PRICE UPDATE ALGORITHM

In this section we devise a price update algorithm that is invoked regularly on the server (e.g., once a day), with the goal to move prices towards the buffer equilibrium over time. Our algorithm is oriented at the tâtonnement process as defined by Walras [15]. However, Walras' algorithm only allowed trades at equilibrium prices. In our system, however, we must allow trades at all times, even out of equilibrium.

6.1 The Algorithm

Because users' preferences are homogeneous of degree zero, collinear price vectors are equivalent. Thus, instead of searching for the equilibrium price vector in $\mathbb{R}^3$, we can simplify the task by looking at projective space $\mathbb{RP}^2$:

$$\mathbb{RP}^2 := \big((p_{\overline{S}}, p_{\overline{U}}, p_{\overline{D}}) \in \mathbb{R}^3 \setminus \{0\}\big) / \sim,$$
$$\text{with} \quad (p_{\overline{S}}, p_{\overline{U}}, p_{\overline{D}}) \sim \lambda(p_{\overline{S}}, p_{\overline{U}}, p_{\overline{D}}) \quad \forall \lambda \in \mathbb{R}^+$$

Thus, we can fix the price of an arbitrary good (the numeraire) and normalize the price vector accordingly. Here, we normalize the price of storage space to 1:

$$p = (p_{\overline{S}}, p_{\overline{U}}, p_{\overline{D}}) \sim (1, \frac{p_{\overline{U}}}{p_{\overline{S}}}, \frac{p_{\overline{D}}}{p_{\overline{S}}})$$

In Section 5.3, we have reduced the problem of finding the buffer equilibrium to finding the root of the function $g(p) = \big(g_{\overline{U}}(p), g_{\overline{D}}(p)\big)$ where

$$g_{\overline{U}}(p) = \Big(\frac{\overline{X}_{\overline{U}}}{\overline{X}_{\overline{S}}} - \frac{f_{\overline{U}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}\Big) \quad \text{and} \quad g_{\overline{D}}(p) = \Big(\frac{\overline{X}_{\overline{D}}}{\overline{X}_{\overline{S}}} - \frac{f_{\overline{D}}^{-1}(Y)}{f_{\overline{S}}^{-1}(Y)}\Big).$$

This formulation of the buffer equilibrium is also useful for the price update algorithm, because finding the root of a function is a well-understood mathematical problem. *Newton's method* is probably the best-known root-finding algorithm, which also converges very quickly in practice. However, it requires the evaluation of the function's derivative at each step. Unfortunately, we don't know the function $g(\cdot)$ and thus cannot compute its derivative. Instead, we only get to know individual points in each iteration and can use these points to estimate the derivative. This is exactly what the *secant method* does for a one-dimensional function.

The problem is that $g(p)$ is 2-dimensional, and thus the secant method is not directly applicable. The appropriate multi-dimensional generalization is *Broyden's method* [4], a quasi-Newton method. Unfortunately, that method requires knowledge of the Jacobian, which we don't know and also cannot even measure approximately. However, we show that one can use an approximation to the diagonal sub-matrix of the Jacobian instead of the full Jacobian matrix. The diagonal sub-matrix of the Jacobian can be approximated by studying changes in the function $g(p)$. This leads to the following quasi-Newton method for multiple dimensions:

Definition 5. (The Price Update Algorithm)

$$p_l^{t+1} = \begin{cases} 1 & \text{for } l = \overline{S} \\ p_l^t - \frac{p_l^t - p_l^{t-1}}{g_l(p^t) - g_l(p^{t-1})} \cdot g_l(p^t) & \text{for } l = \overline{U}, \overline{D} \end{cases}$$

For the implementation of the price update algorithm in our system we took care of a few special cases (e.g., exactly reaching the equilibrium such that terms cancel out). Due to space constraints, we omit the details here.

6.2 Theoretical Convergence Analysis

In this section we prove convergence of the price update algorithm under quite mild assumptions. We begin with the analysis of the convergence of the following iteration rule:

$$x^{(k+1)} = x^{(k)} - D(x^{(k)})^{-1} F(x^{(k)}) \tag{29}$$

where F is a function $F : \mathbb{R}^n \to \mathbb{R}^n$ and D is the diagonal sub-matrix of the Jacobian J of F. We define the matrix L by the rule $J(x) = D(x) + L(x)$, i.e., L comprises of the off-diagonal partial derivatives in the Jacobian. For this iteration rule, the following theorem holds (proof is omitted due to space constraints):

THEOREM 3. *Let F be a continuously differentiable function. Suppose that in the iteration rule given by equation (29), $x^{(0)}$ is chosen close enough to a root x^* of F, $J(x^*)$ is non-singular, J and D are Lipschitz continuous, and $L(x^*) = 0$. Then the successive iterations $x^{(k)}$ produced by the iteration rule converge to x^*, and the rate of convergence is at least Q-linear.*[6]

The problem one faces when trying to apply the secant method to higher dimensions is that the system of equations provided by $J_k \cdot (x^{(k)} - x^{(k-1)}) \simeq F(x^{(k)}) - F(x^{(k-1)})$ (where J_k is the current estimate of the Jacobian) is under determined. However, if one uses the diagonal approximation to the Jacobian, then the system is fully determined. What Theorem 3 says is that under certain conditions, using the diagonal sub-matrix of the Jacobian instead of the full Jacobian in the given iteration rule, still leads to convergence to a root of the function.

Equipped with Theorem 3, it is now easy to prove that the price update algorithm given in Definition 5 converges to a buffer equilibrium. We only need to consider the update algorithm for resource prices $p_{\overline{U}}$ and $p_{\overline{D}}$ because the price for space remains constant at 1. Consider the function $g(\cdot)$, and as before, J is the Jacobian of $g(\cdot)$, D is the diagonal sub-matrix of J, and L is defined by the rule $J(x) = D(x) + L(x)$.

COROLLARY 2. *Consider the price update algorithm given in Definition 5. If $g(\cdot)$ is a continuously differentiable function, $p^{(0)}$ is chosen close enough to a root p^* of $g(\cdot)$, the Jacobian $J(p^*)$ is non-singular, J and D are Lipschitz continuous, and $L(p^*) = 0$, then the price update algorithm converges to an equilibrium price vector p^*, and the rate of convergence is at least Q-linear.*

[6]We can in fact prove that the iteration rule exhibits faster than Q-linear convergence: just like Broyden's method, its convergence is locally Q-superlinear. However, showing this result requires a more intricate argument and we defer this to a future extended version of this paper.

PROOF. We have shown in Section 5.3 that if we find a price vector p^* such that $g(p^*) = 0$, then we have reached a buffer equilibrium. Thus, we only have to show that the price update algorithm converges to a root of the function $g(\cdot)$. Now, note that the price update algorithm provided in Definition 5 defines a quasi-Newton iteration rule that uses the diagonal sub-matrix of the Jacobian of the function $g(\cdot)$, equivalent to the iteration rule given in equation (29). By Theorem 3, that iteration rule converges locally to a root of $g(\cdot)$, and the rate of convergence is at least quadratic. □

One might wonder how restrictive the conditions of Theorem 3 and Corollary 2 are. The condition that the matrices J and D be Lipschitz continuous puts upper bounds on how fast the partial derivatives of the function can change. It turns out that one can relax this assumption to just that of J and D being Lipschitz continuous in a neighborhood of the root without affecting the conclusions of the theorem and corollary. Local Lipschitz continuity near the neighborhood of the root seems like a plausible condition for $g(\cdot)$ to satisfy because it is hard to envision wild changes in the function near an equilibrium point. The non-singularity of $J(p^*)$ means that our function does not have a higher order zero at the equilibrium point. If this assumption fails, then the Newton method is known to converge only linearly to the root. It is likely that our algorithm would still converge, but we do not have a proof of this. The local convergence of our method is an aspect we share with all Newton's methods operating in multiple dimensions, and this is the most worrisome property as well as the hardest to get a handle on. If $\|J(p^*)^{-1}\|$ and Lipschitz constants of J and D around p^* are all small, then the basin of convergence is large. However, it seems that only experimental evidence can validate whether this assumption is reasonable in our situation.

6.3 Experimental Convergence Analysis

In this section we analyze the convergence of the price update algorithm experimentally via simulations. To make sure a unique buffer equilibrium exists, we must specify utility functions that satisfy the properties of Theorems 1 and 2. It is well known that the CES (constant elasticity of substitution) utility function with the following form $u(x_1, x_2) = (\alpha_1 x_1^\rho + \alpha_2 x_2^\rho)^{\frac{1}{\rho}}$ satisfies the gross substitutes condition for $0 < \rho < 1$ (cf. [11], p 612). Furthermore, the utility function $u(x_3, x_4) = \min(\alpha_3 x_3^\tau, \alpha_4 x_4^\tau)$ implies that x_3 and x_4 are perfect substitutes (while still maintaining strictly convex preferences). Thus, a whole class of utility functions satisfying the necessary assumptions is given by:

$$u_i(K_i, Y_i) =$$
$$(\alpha_1 K_{\overline{S}}^\rho + \alpha_2 K_{\overline{U}}^\rho + \alpha_3 K_{\overline{D}}^\rho)^{\frac{1}{\rho}} + \min(\beta_1 Y_B^\tau, \beta_2 Y_\Sigma^\tau, \beta_3 Y_R^\tau)$$

Because this utility function is too complex to derive supply and demand functions analytically, we used the NLP-solver MINOS which is able to solve the utility maximization problem via reduced-gradient methods very efficiently.

Experimental Set-up. We implemented a market simulation with 100 agents, each initialized with a different utility function. For each agent we randomly picked an endowment vector w_i, a current demand vector y_i, and the parameters $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, \rho$ and τ. After each time step, we used MINOS to solve 100 utility maximization problems and fed the new supply and demand vectors back into the simulation to calculate the next price iteration. We stopped the experiment once $|g_U + g_D| \leq 0.00001$.

Experimental Results. Our findings from this experiment confirm the results from the theoretical convergence analysis. When the initial price vector was chosen close enough to the equilibrium price vector, then the price update algorithm converged very quickly. However, when a price vector far away from the equilibrium prices was chosen, the algorithm sometimes never converged, which was expected. While these preliminary results are encouraging, we are are much more detailed experimental analysis of the price update algorithm in the future. For example, we will study how the algorithm reacts to demand and supply shocks, what happens when the users update their settings with different delays, what happens when a subset of the users becomes price-insensitive (cf. [13], discussion about price discoverability), or what happens if individual users suddenly drop out of the market or join the market.

7. CONCLUSION

In this paper, we have presented the design and analysis of a novel resource exchange market underlying a P2P backup application. At all times, for the model formulation and the theoretical analysis, the focus was on the actual implemented system which we have successfully tested in alpha version. During the design phase for this market, we followed the *hidden market design* paradigm [14]. The result is a system that hides many of the market complexities and allows the users to interact with the market in a very indirect way. In contrast to existing work in this area where users are generally required to constantly update their supply and demand, in our model users choose bounds on their maximum supply in return for being allowed to consume a certain maximum amount of backup services in the future. In this setting, we have proved the existence and uniqueness of a buffer equilibrium under reasonable assumptions. We have introduced a price update algorithm and have shown that it converges Q-linearly, given that initial prices are chosen close enough to the equilibrium. However, we are planning a more extensive experimental analysis to fully understand the behavior of the algorithm. The most surprising result is the limited controllability of the size of the supply side buffer in equilibrium. We have shown that supply and demand prices would have to be decoupled to enable more control.

In future research, we will study how to design new UIs that allow for this decoupling without presenting false information to the users. In ongoing work we are augmenting the market design with a payment mechanism that will provide for robustness against strategic deviations from users that manipulate the protocol by reprogramming their software client. Furthermore, we are extending our current design to allow for real monetary payments. On the one side, users will then be able to pay for their consumption of services by either providing their own resources or by paying with real money, and on the other side, users will then also be able to earn real money by supplying their resources.

We believe that the market design presented in this paper has applicability beyond P2P backup systems. For example, significant research efforts currently go into the development of smart grids [6]. To effectively involve the individual consumers of electricity in these new energy markets, the development of new user interfaces and hidden market designs will be necessary. Hopefully, the ideas we presented in this paper will inspire other researchers to develop similar market designs for novel applications in many other domains.

Acknowledgements

We are thankful to David Parkes, Kamal Jain, and Alex White for very helpful discussions on this work. Furthermore, we thank seminar participants from Harvard EconCS, HBS, Humboldt University Berlin, and NetEcon'09 for very useful feedback. Seuken gratefully acknowledges the support of a Microsoft Research PhD Fellowship.

8. REFERENCES

[1] J. Alexander S. Kelso and V. P. Crawford. Job Matching, Coalition Formation, and Gross Substitutes. *Econometrica*, 50(6):1483–1504, 1982.

[2] C. Aperjis and R. Johari. A Peer-to-Peer System as an Exchange Economy. In *Proceedings from the Workshop on Game Theory for Communications and Networks (GameNets)*, Pisa, Italy, October 2006.

[3] W. J. Bolosky, J. R. Douceur, and J. Howell. The Farsite Project: A Retrospective. *SIGOPS Operating Systems Review*, 41(2):17–26, 2007.

[4] C. G. Broyden, J. E. Dennis Jr., and J. J.Moré. On the Local and Superlinear Convergence of Quasi-Newton Methods. *J. Inst. Math. Appl.*, 12:223–245, 1973.

[5] L. P. Cox and B. D. Noble. Samsara: Honor Among Thieves in Peer-to-Peer Storage. In *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP)*, pages 120–132, Bolton Landing, NY, 2003.

[6] U. S. D. O. Energy. Grid 2030: A National Vision for Electricity's Second 100 Years. 2003.

[7] M. J. Freedman, C. Aperjis, and R. Johari. Prices are Right: Managing Resources and Incentives in Peer-Assisted Content Distribution. In *Proceedings of the 7th International Workshop on Peer-to-Peer Systems*, Tampa Bay, FL, February 2008.

[8] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[9] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman. Tycoon: An Implementation of a Distributed, Market-based Resource Allocation System. *Multiagent Grid Systems*, 1(3):169–182, 2005.

[10] J. Li and C. Zhang. Distributed Hosting of Web Content with Erasure Coding and Unequal Weight Assignment. In *Proceedings of the IEEE International Conference on Multimedia Expo*, pages 27–30, Taipei, June 2004.

[11] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[12] S. Seuken, D. Charles, M. Chickering, and S. Puri. Market Design and Analysis for a P2P Backup System. In *Proceedings of the Workshop on the Economics of Networks, Systems, and Computation (NetEcon)*, Stanford, CA, July 2009.

[13] S. Seuken, K. Jain, D. Tan, and M. Czerwinski. Hidden Markets: UI Design for a P2P Backup Application. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, Atlanta, GA, April 2010.

[14] S. Seuken, D. C. Parkes, and K. Jain. Hidden Market Design. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, Atlanta, GA, July 2010.

[15] L. Walras. *Eléments d'économie politique pure; ou, théorie de la richesse sociale (Elements of pure economics; or, the theory of social wealth)*. Corbaz, Lausanne, 1874.

[16] F. Ygge and H. Akkermans. Power Load Management as a Computational Market. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS)*, pages 393–400, Kyoto, Japan, 1996.