Feedback-based Cooperative Content Distribution for Mobile Networks

Mikael Lindberg
Department of Automatic Control
Lund University
mikael.lindberg@control.lth.se

ABSTRACT

A feedback-based scheme for cooperative content distribution for mobile systems aimed at reducing energy and licensed spectrum usage is presented. The paper is motivated by the problems in implementing peer-to-peer based content distribution in mobile networks due to the risks of unfair energy expenditure. A technique for incentivizing cooperation and for enforcing agreements between peers is discussed as well as the dynamics of the resulting barter-like economy. Rationale for a heuristic solver is provided together with simulation-based analysis of expected savings. The scheme is evaluated for cases with deterministic as well as random initial conditions.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

Keywords

Cooperative, content distribution, peer-to-peer, feedback, optimization

1. INTRODUCTION

As mobile networking grows increasingly pervasive, the limited resources are becoming the focus of much debate. A recent FCC technical report [16] projects that the US will run out of radio spectrum as early as 2013 and battery life is a constant concern when designing mobile phones and pads. Reducing the load on the mobile network infrastructure, such as base stations and backhaul links, is therefore essential in order to support the demands of coming years.

An especially relevant scenario is the distribution of music, video and applications from centralized services, such as Google Play or the Apple iTunes Store. When a popular artist releases a new album or when a commonly used application is updated, it can be assumed that repositories will see a significant rise in requests for those data objects. That many clients requests the same information suggests that there could be something to gain from cooperation among peers.

While peer-to-peer assisted mechanisms [4][12] are commonly used in content distribution schemes to off load network links and central repositories, they are still rarely applied in the mobile case out of concern for the energy consumption involved. While the feedback mechanisms used in e.g. BitTorrent are designed to reward cooperation, the transient nature of mobile networks can deny a client from being able to benefit from accumulated good-will. There is also the risk of "leeching", that is clients taking advantage of other peers without contributing anything in return.

For a peer-to-peer solution targeted at mobile networks to make sense, it will have to take the limited energy resources of a mobile phone client into account. It will also have to provide guarantees that cooperation will pay off within a time period relevant to the client and protect against leeching. As shown by game theory [6], lack of trust between parties will prevent cooperation from taking place.

Trust is however not enough, there must also be some form of benefit from participating in the cooperative scheme. Providing means to reduce energy consumption, a premium resource for mobile clients, seems like an attractive strategy.

This paper shows how cost savings, both in terms of energy and long distance traffic, can be used to incentivize cooperation between mobile clients and how a mechanism for enforcing agreements can be implemented in order to establish trust between parties. The model proposed here has similarities with bartering economics [14], a type of trade often used where there are no trustworthy forms of currency.

The scenario used as a motivating example is the distribution of files to a population of mobile clients, for example a firmware upgrade or a new version of a popular application. The clients involved are assumed to have a primary connection to a remote service offering this file, reachable through an expensive long distance link (e.g. 3G or 4G), and the capability for local connections with co-located clients using cheaper communication forms (e.g. WLAN or Bluetooth).

1.1 Outliner of the paper

Section 3 discusses the issues with making clients cooperate in an environment where they could be taken advantage of, how to incentivize cooperation, and outlines a mechanism to enforce agreements. A model of the resulting system is introduced in Section 4. Section 5 discusses the dynamics of fair exchanges and introduces the swap graph as a tool

PM2HW2N'13, November 3–8, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2512840.2512855.

for discovering opportunities for cooperation. A baseline algorithm for arbitrating mutually beneficial exchanges is discussed in Section 6, but due to its significant computational requirements, a heuristic alternative is presented in Section 7. The algorithm is then applied to a small example to illustrate some of its properties in Section 7.4 and 7.5. Section 8 studies how the size of the problem affects the solution, how it can be decomposed into subproblems and compares the results with those obtained through the method in [17]. The assumption on initial state is relaxed in Section 9. Sections 10 and 11 summarize the findings and discuss some ideas for future extensions to this work.

2. RELATED WORKS

Mobile implementations of peer-to-peer schemes have been considered for some time now, though initially primarily as a way to distribute content in pure ad-hoc networks [11] [7] and not as an energy conservation measure.

Later publications, such as [15] [17] [16], are more concerned with the resource related challenges specific to mobile peer-to-peer networking. Energy and spectrum usage is a common concern, though the focus is typically on the population as a whole or the average case, rather than the individual.

Exploiting multi-mode communications to save energy is discussed as an off-line optimization problem in [17], which uses a detailed energy model and a game theoretic approach to derive an optimal dissemination policy. Fairness is explicitly discussed, but the method does not account for changing populations or how to deal with leeching.

Barter-like economics arise in a variety of seemingly unrelated fields, such as kidney exchanges [5] or apartment contract trading [14]. These particular examples differ from the content distribution scenario in that there is no central repository (i.e. cooperation is the only option) and that once a trade has occurred, clients will withdraw from the system. The graph structure used to find chains of trades in [5] is analyzed through stochastic methods that could be adapted to analyzing data object trades.

3. INCENTIVIZING COOPERATION

It can be shown through game theory that two rational decision makers will choose not to cooperate if there is the possibility of one of them taking advantage of the other. This case is often referred to as the "Prisoner's Dilemma" [6] and illustrates the importance of trust between parties in a voluntary cooperation scenario.

Translated into the terms of a cooperative file retrieval scenario, if two geographically co-located mobile clients, A and B, seek to retrieve two data objects both clients want from a remote service (typically a file server), they could could potentially cooperate by sharing the cost, in terms of energy or traffic fees, for long distance traffic.

Let w_l denote the cost of accessing the data from a remote source and w_s the cost of communicating with a co-located source. Accessing data requires energy expenditure by both sender and receiver, meaning that one local access incurs a cost of w_s for both parties. The energy expended by the remote server is not accounted for in this example. Assume also that $w_l \gg w_s$.

A and B can now independently choose to either be cooperative, that is, allowing the other party to copy a data object, or uncooperative, that is, not giving the other client access. The outcome of the possible scenarios in terms of cost to retrieve both objects are listen in the table below.

Cost for (A,B)	B cooperative	B uncooperative
A cooperative	$(w_l + 2w_s,$ $w_l + 2w_s)$	$(2w_l + w_s,$ $w_l + w_s)$
A uncooperative	$(w_l + w_s,$ $2w_l + w_s)$	$(2w_l, 2w_l)$

If both choose to cooperate, the costs will consist of one repository access to fetch one object and then two short range accesses, one to deliver the object to the peer and one to fetch the other object from the same. In case only one chooses to cooperate, both will start out fetching one object from the repository, but only one will be able to fetch from the peer. The other will then have to perform another repository access to complete the set.

The Nash equilibrium [6] of this game is that both A and B choose to be uncooperative. Two principle strategies can be seen that would resolve this problem, either

- w_s must be reduced to 0, thereby making it "free" to risk cooperation or
- the off-diagonal choices must be eliminated, making it impossible for one party to exploit the other.

The first strategy models the behavior of traditional peer-to-peer networks, where sharing is considered to be without cost [4]. Even if a participant is taken advantage of the majority of the time, the occasional win is achieved at no cost. In the mobile setting, the risk of running out of energy or loosing contact with surrounding parties makes such assumptions unrealistic.

It is therefore necessary that the exchange system provides potential participants with guarantees that the benefits outweigh the costs. In the general case, this would require appropriate models for user behavior, which is outside the scope of this paper. A simpler scenario can be achieved by requiring that all transactions are bilaterally either cooperative or uncooperative, thereby effectively resorting to the second strategy above.

The question of whether or not to allow multicast transfers is problematic. While publications have shown that this improves the efficiency of cooperation[16][17], there is the risk that clients will opt out of the exchange system and just listen to multicasts, essentially re-introducing a Prisoner's Dilemma like situation. This work therefore assumes that all communication is unicast between two parties.

Furthermore, it is assumed that an exchange system can enforce the adherence to agreements between clients, which will in the case of data object exchanges be referred to as fair exchanges. Exactly how this is done is of less importance to the results presented, but for the sake of feasibility a prototype method is outlined below.

3.1 Prototype contract mechanism

Assume a set of parties C have agreed to exchange a set of objects D according to some scheme. The agreement, or contract, in the form of a list of tuples denoting (supplier, receiver, object) $\in C \times C \times D$, is handed over to an exchange system E, a physical 3rd party in the form of a remote service. E creates a set of encryption keys, one for each unique object in the agreement. The suppliers of each

object are then given the corresponding keys, which they then use to encrypt the objects, after which they transmit them. When all receivers have signaled to E that they have indeed received the complete transmissions, E sends out the appropriate decryption keys to the participants.

4. SYSTEM MODEL

Consider a population of mobile terminals capable of multi-mode communication, that is, able to use several wireless communication standards. Specifically, they support both an expensive form of long range communication (e.g. 3G in the form of UMTS or W-CDMA) and a less expensive short range alternative (e.g. WLAN or Bluetooth). Building on the notation introduced in Section 3, let $C = \{c_i, i = 1..N_c\}$ denote a subset of the population, such that all are within short range communication distance of each other. Furthermore, let $D = \{d_j, j = 1..N_d\}$ denote a set of data objects that all clients desire to retrieve. These objects could be individual files or parts of one larger file, split into parts to facilitate distribution. For the sake of simplicity, assume all parts are of equal size.

All parts of D are available from a central repository accessible only over long range communication. The nominal cost for a client to download all parts of D is thus $N_d w_l$ but clients can cooperate by trading objects via short range communication and thereby reduce their total transfer cost. Because of the cost associated with sharing a data object with another client, it is assumed that a client will only provide a requested object if it is guaranteed to get an object in return, referred to as a *fair exchange*. This rule is referred to as the *exchange policy*.

The client population C, the target data set D, and the policy P under which trades take place constitute an Exchange System $E = (C, D, P)$, with the objective of allowing clients to minimize their data retrieval costs. The state of the system is the contents of the client side caches that contain the data objects once a client has retrieved it. When an object is exchanged between two clients it is copied, the original remains with the source.

In this formulation E implements a centralized decision mechanism with complete knowledge of the system state. The system dynamics evolve in discrete time steps of indeterminate length, but with the following logical sub-steps *discovery, arbitration* and *effectuation* that are repeated in a loop.

- **Discovery**. During discovery, clients join the exchange system and submit their current state. Let $\kappa(c)$ be a function that returns the data objects currently possessed by the c and $cardinal(\kappa(c))$ a function that returns the number of elements in $\kappa(c)$.
- **Arbitration**. Once the system state is established, the exchange system decides which trades that will occur. Clients not part of a trade will perform a default action, that can be either fetching an object from the central repository or passing (i.e. doing nothing).
- **Effectuation**. Finally all decisions are carried out. The completion of these actions marks the end of the time step, after which next immediately starts with a new discovery phase.

Clients can be expected to join or leave the exchange system from one time step to the next, either voluntarily (e.g. having completed the data set or user command) or involuntarily (e.g. through loss of connection), making repeated discovery necessary. By choosing to pass in the arbitration phase, a client can bide its time, hoping for a more beneficial situation to arise in a future time step.

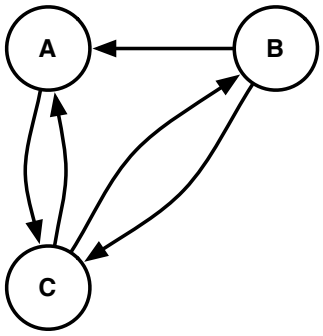

Figure 1: A swap graph for the system $A[1,2], B[1], C[3]$

5. THE DYNAMICS OF FAIR EXCHANGES

The data object exchange scenario differs from many types of bartering economics primarily in that a client is typically interested in several trades rather than just one. It will withdraw from the exchange system once it completes its set of data objects. This removes an attractive cooperation partner, one who possesses all the objects desired by the other clients. This is a key complication that an effective exchange policy must take into account.

To keep the pool of cooperation partners as large as possible, it is important to prevent some from completing their sets far ahead of the rest. Making client states (i.e. the contents of the client side object caches) as diverse as possible will further increase the probability that any one client will find a peer that can provide desired objects, while needing those already in possession.

5.1 The swap graph

To further discuss the properties of these systems, the concept of the swap graph will be used. This is a directed graph representation of what possible trades are possible, where each vertex represents a client and each edge represents a potential object transfer. If client A has an object desired by client B, then the swap graph will contain an edge from B to A, labeled with the object in question, to indicate the dependency. As there can be multiple dependencies between two clients, there can be multiple edges but with different labels.

As an example, consider a case with the client set $\{A, B, C\}$ and the data object set $\{1, 2, 3\}$. In the example, let client A possess objects 1 and 2, represented by the short hand notation $A[1, 2]$. Assume now that the total system state is $A[1, 2]$, $B[1]$, $C[3]$. The corresponding swap graph is seen in Figure 1. Possible fair exchanges are seen as cycles in the graph, with in total four in the example, as detailed in Figure 2. In this case they are mutually exclusive, which leads to the central question of arbitration, that is determining what exchanges should take place in order to optimize the objectives?

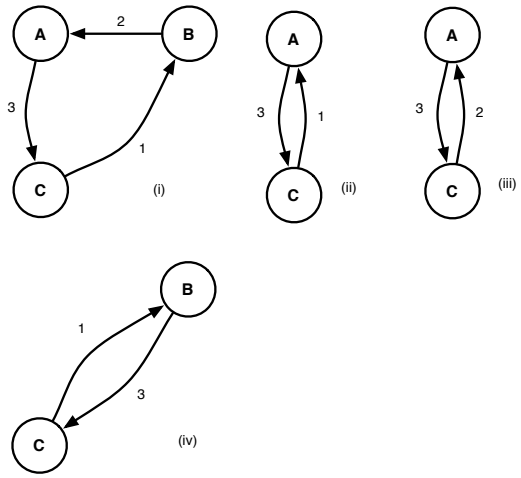

Figure 2: The cycles given by the swap graph in Figure 1. Each of the cycles are mutually exclusive, meaning only one of them can take place. Arbitrating this conflict is a responsibility of the exchange system.

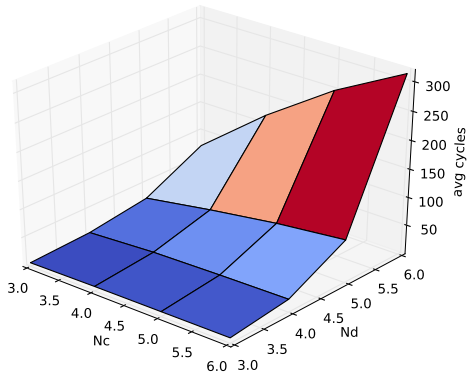

Figure 3: The number of cycles grows very fast with the dimensions of the system. This plot shows the average number of cycles over 50 simulations when all clients are assigned randomly chosen states.

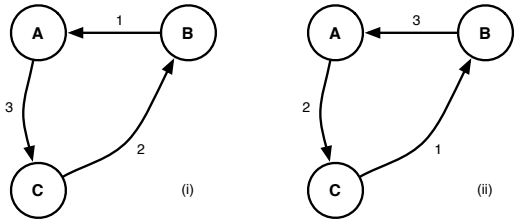

Figure 4: Swap graphs for the special case discussed in Section 5.2. The system is initialized with the state $A[1], B[2], C[3]$, as shown in Graph (i), which allows for a 3-way exchange involving all clients, leading to the situation depicted in Graph (ii).

In its entirety, the problem is a multistep decision problem, where in each step determining the set of exchanges to perform involves finding the best set of non-overlapping cycles in the graph, a version of the classic NP-hard maximum set packing problem [10]. The computational complexity in each step grows as $2^{N_{cycles}}$ and since the number of cycles grows very fast with set sizes, as shown in Figure 3 , finding the globally optimal solution is intractable for realistic scenarios involving hundreds of clients. However, a possible key to alternative strategies presents itself by studying a special case of $N_c = N_d$.

5.2 $\mathbf{N_d = N_c}$

Consider a case with $N_c = N_d = 3$, with the system state A[1], B[2], C[3] and the corresponding swap graph shown in Figure 4-i. After trivially selecting the exchanges, the state becomes A[1,3], B[1,2], C[2,3], with the swap graph in Figure 4-ii. This gives another trivial decision that ends the scenario (as all clients are done) with an optimal cost for all clients.

The two following observations can now be made:

1. The trivial optimal strategy above is always possible when all clients have the same number of objects and every object occurs the same number of times in the system.

2. The solution is not unique in general, there might be many other ways to achieve the same optimal global cost.

Let

$$n_c = cardinal(\kappa(c))$$

and

$$f_d = \sum_{c \in C} I_d(c)$$

where $I_d(c)$ is an indicator function defined as

$$I_d(c) = \begin{cases} 0 & \text{if } d \notin \kappa(c) \\ 1 & \text{if } d \in \kappa(c) \end{cases}$$

Furthermore, let

$$\bar{n} = \frac{1}{N_c} \sum_{c \in C} n_c \text{ and } \bar{f} = \frac{1}{N_d} \sum_{d \in D} f_d$$

Using this notation, the condition from Observation 1 can be formalized into

$$n_i = n_j, \forall i, j \in C \text{ and } f_k = f_l, \forall k, l \in D \quad (1)$$

from here on referred to as Condition A.

Consider now the function

$$J = \sum_{c \in C} (n_c - \bar{n})^2 + \sum_{d \in D} (f_d - \bar{f})^2 \quad (2)$$

The quantity J can be seen to denote the distance to A, or if it is assumed that the optimal trajectory will be followed once A is fulfilled, the distance to the optimal trajectory.

The quantities $\sum_{c \in C}(n_c - \bar{n})^2$ and $\sum_{d \in D}(f_d - \bar{f})^2$, essentially the sample variance of the client cache sizes and object frequencies respectively, can be interpreted to model two aspects of how well the exchange system will work.

If the cache size variance is high, then some clients will finish way ahead of others, thereby removing many objects from the system. It therefore makes sense to prioritize clients with few objects when arbitrating exchanges.

If the frequency variance is high, some objects are rare, meaning few clients can offer them, while some are frequent, meaning few clients want them. Both cases will lead to fewer possible exchanges involving these objects. It therefore makes sense to try to keep object frequencies uniform.

6. BASELINE ALGORITHM

Using Equation (2), it is possible to formulate a one-step decision algorithm based on minimizing J. Basing decisions on only the currently measurable state of the system, in this case the contents of the client side caches, is a feedback control approach. This has the advantage of being robust to disturbances, such as failed transfers or clients arriving to or departing from the exchange system. A pre-calculated multistep decision strategy would, on the contrary, have to be recalculated if for instance the state of the system suffers an unforeseen perturbation, such as a failed object transfer or clients leaving E.

Let X denote the system state, u denote a set of exchange agreements to carry out and $J(X|u)$ denote the cost function evaluated for the state after X has been subjected to u. Furthermore, let $\rho(X)$ be a function that maps the system state to a set of possible exchange agreements. The feedback arbitration policy can now be written as

$$u = \arg\min_{u \in \rho(X)} J(X|u) \quad (3)$$

Because of the combinatorial nature of the optimization problem used to calculate (3), designing the function $\rho()$ is non-trivial. The formulation is very close to the maximum set packing problem and as discussed in Section 5.1, the number of possible decisions grow unmanageably large even for modestly sized problems. However, it can still useful to compare other solvers with the result given if $\rho()$ is assumed to generate all possible agreements, hereon referred to as the *baseline algorithm*.

7. HEURISTIC SOLVER

In order to further study this type of exchange system, a simple heuristic solver has been developed. Its main characteristics are that

- it is deterministic, that is, a given state always yields the same decision,
- it is guaranteed to find at least one exchange unless there are none, which is trivial to test for, and
- it is computationally cheap.

7.1 Heuristic cycle finding

The heuristic solver uses a steepest decent style graph walking method for finding cycles. Let

$$J_c(X) = \frac{\partial J(X)}{\partial n_c} \quad (4)$$

$$J_d(X) = \frac{\partial J(X)}{\partial f_d} \quad (5)$$

and let G' be the *pruned swap graph*, obtained by repeatedly searching G for nodes with zero in-degree or zero out-degree [8], which obviously cannot be part of a cycle, stopping when no more nodes can be removed. It can easily be seen that the all connected subgraphs in G' contain at least one cycle. As all nodes have outgoing edges with no edges to itself, there are no "dead ends" in the graph and since the graph is finite a walk must eventually end up in a node that has been visited before.

Because there can be multiple edges between nodes, the algorithm must keep track of both visited nodes and edges. This is done with a data structure called *trace*, a list of alternating node and edge elements.

A pseudocode representation of the algorithm used for finding cycles is then given as Algorithm 1. `Sort()` orders elements in a list according to their unique text labels, which is done in order to make the algorithm deterministic. `FirstElement()` and `LastElement()` returns the first and last elements of a list respectively and `List()` creates a list from the arguments. The function `gradJ` calculates $J_c(X) + J_d(X)$, using for c the end node of the edge and for d the data object associated with the edge. The last operation removes the preamble of the trace, as the initial parts might not be part of the cycle.

7.2 A heuristic $\rho()$

Building on the heuristic cycle finder, a heuristic $\rho()$, denoted $\rho_H()$, can then be defined. Let `HeuristicFind` be the heuristic cycle finder algorithm defined in Section 7.1. A pseudocode representation of $\rho_H()$ is shown as Algorithm 2.

The exchange agreements to be carried out are generated through repeatedly searching for cycles in G and removing the nodes in found cycles until the remaining graph is empty.

7.3 Default actions

The clients not part of any exchange are still able to act, though their actions are limited to either

- fetching an object from the remote repository, or
- passing (i.e. doing nothing).

The decision comes down to if the client is willing to wait and see if a better situation arises or if it should instead pay for immediate access to a data object. In this work, this is modeled by a client parameter named *skipcount* that decides how many times a client that is not part of any exchange agreement after arbitration is willing to wait, essentially a

```
Input: pruned swap graph G'
Output: a cycle in G'
begin
    current ←
    FirstElement(Sort(GetNodes(G')))
    trace ← List(current)
    done ← false
    while not done do
        Es ← Sort(GetOutEdges(current))
        Gs ← List()
        mingrad ← MaxFloat()
        foreach e in Es do
            v ← SourceNode(e)
            d ← DataObject(e)
            u ← ReceiverNode(e)
            if GradJ(v, d, u) < mingrad then
                next ← v
                mingrad ← GradJ(v, d, u)
                append d to trace
            end
        end
        if next in trace then
            done ← true
        else
            current ← next
        end
        append next to trace
    end
    while FirstElement(trace) ≠
    LastElement(trace) do
        remove first element from trace
    end
    return trace
end
```

Algorithm 1: Pseudocode representation of the heuristic cycle finder.

form of time out. An important special case arises when a client enters the exchange system with an empty cache. It will then have to download at least one object from the remote repository in order to have something to trade with.

As the other parts of the algorithm aim to minimize J, it would seem appropriate that the default actions do the same. This can be most easily done by simply fetching the least frequent object currently not already possessed by the client. In order to preserve determinism, the candidates are sorted by frequency first and label second.

7.4 A 10 x 10 example

The performance and behavior of the algorithm have been studied through simulations in an environment built in Python [3] using off-the-shelf modules for graph algorithms [2] and plotting [1]. The source is provided separately [13].

Consider a scenario with 10 clients and 10 data objects, where all clients start out empty. Assume that $w_l = 1$ and that w_s is sufficiently small that it can be approximated to 0 under a policy of fair exchanges. Figure 5 shows how J, the total communication cost for the entire system and the worst case individual client communication cost evolve over time, stopping when all clients have all objects. Communication costs are normalized so that the nominal case where all objects are fetched from the remote repository corresponds to

```
Input: A swap graph G
Output: A list of traces representing exchange
        agreements
begin
    u ← List()
    done ← false
    while not done do
        G' ← Prune(G)
        upart ← List()
        if not G' empty then
            upart ← HeuristicFind(G')
        else
            done ← true
        end
        remove nodes in upart from G
        append upart to u
    end
    return u
end
```

Algorithm 2: Pseudocode representation of the heuristic decision function $\rho_H()$ where `HeuristicFind()` is a call to Algorithm 1.

a cost of one. A plot showing how many clients are involved in trading in each step is also provided. The heuristic solver is used and clients use a `skipcount` of 0.

As the initial state satisfies Condition A, the optimal trajectory is known and would give a worst case individual cost of one remote access, resulting in a normalized individual cost of $1/N_d = 0.1$, and a normalized total cost also of $1/N_d = 0.1$. The heuristic solver is not able to achieve these costs, but manages to reduce the total cost by 90% and the worst individual cost with 60%, compared with the non-cooperative case.

7.5 Influence of skipcount

By varying the skipcount parameter, it is possible to make a tradeoff between cost and latency. As expected, increasing the skipcount generally decreases the resulting total cost, but finding the point after which increasing it further provides no benefit has so far only been done experimentally. Figure 6 shows a repetition of the setup from Section 7.4, but with a skipcount of 2. In this case the optimal cost is nearly achieved, but the scenario takes more time steps, as can be seen in the last time step where the individual cost increases by one remote access. The total cost also increases but negligibly so.

8. SET SIZES AND PROBLEM DECOMPOSITION

The problem lends itself to decomposition into parts, as evident when studying the results in Figure 7. This shows the worst case individual cost for different combinations of N_d and N_c (using the heuristic solver and a skipcount of 10 in all cases) and it can be seen that the level jumps approximately each time N_d crosses a multiple of N_c. This can be explained by thinking of the scenario as a combination of sub-scenarios.

8.1 Case 1: $\mathbf{N_d < N_c}$

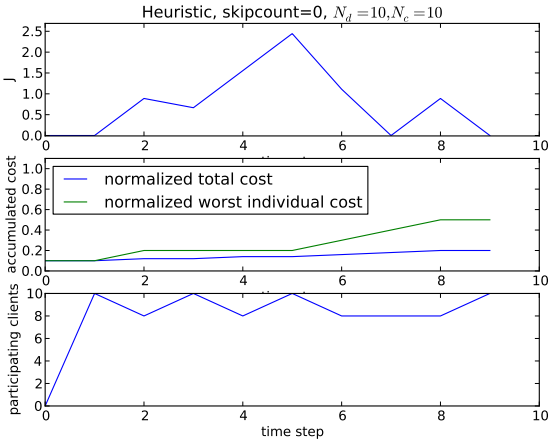

Figure 5: System trajectories for the example in Section 7.4. The costs have been normalized so that a cost of 1 corresponds to the worst case cost, that is the case where all objects are fetched from the remote repository. For the individual cost the normalization factor is $1/(N_d w_l)$ and for the total cost the factor is $1/(N_c N_d w_l)$.

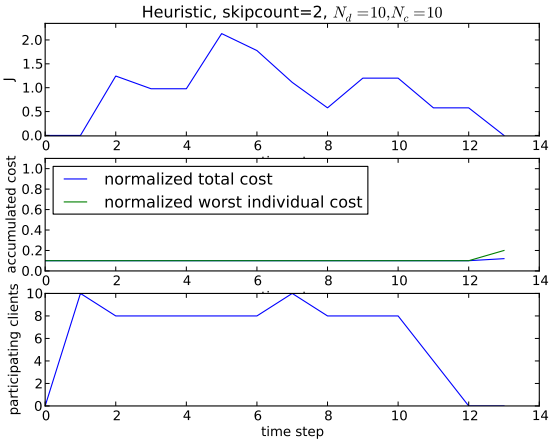

Figure 6: System trajectories for the example in Section 7.5, with the `skipcount` parameter set to 2, resulting in lower individual and total costs at the expense of more time steps. Only a single client is forced to do two repository accesses, which is seen in the slight increase of worst case cost in the final time step.

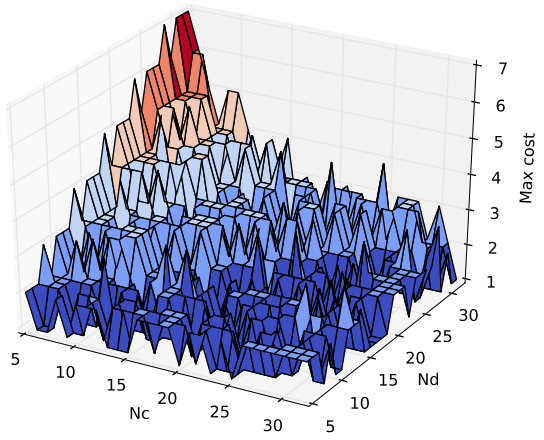

Figure 7: The worst case individual costs (non-normalized) for scenarios of various sizes, all using `skipcount` of 10 which has experimentally been proven sufficient for all the cases in this simulation to reach their lowest costs. w_l is set to 1 (i.e. the cost is the same as the number of repository accesses).

If there are more clients than data objects, the clients can divide themselves into groups, ideally of size N_d, and apply the nominal strategy in parallel. Each client should only need to pay for one repository access.

8.2 Case 2: $\mathbf{N_d > N_c}$

If there are more data objects than clients, then the data objects can be divided into groups, ideally of size N_c and handled in serial manner. Each client must pay for one repository access per serial group.

From this it can be concluded that

$$\lceil N_d/N_c \rceil \tag{6}$$

is a reasonable predictor for the worst case individual number of remote accesses using this solver or with words, a group of N_c clients can often cooperate around a set of N_d objects allowing each client to only pay for once remote repository access. If the remainder is non zero, some clients are likely to be prohibited from trading for one object, thereby raising the worst case number of accesses by one. Because of the above listed decomposition properties of the problem, (6) is also the lower bound on the worst case communication cost.

8.3 Comparison with a feed forward approach

The method presented in [17] shows results with energy savings from approximately 60% (unicast case) to 95% (multicast case), when sufficiently many clients are involved. The results are not immediately comparable with those presented in this paper, as the energy consumption of the IEEE 802.11b WiFi traffic is not sufficiently low to justify the assumption that $w_l \gg w_s$. However, the newer IEEE 802.11n standard has been shown to be in the order of a factor 10 times as efficient in J / bit as 802.11b [9], making the assumption more realistic.

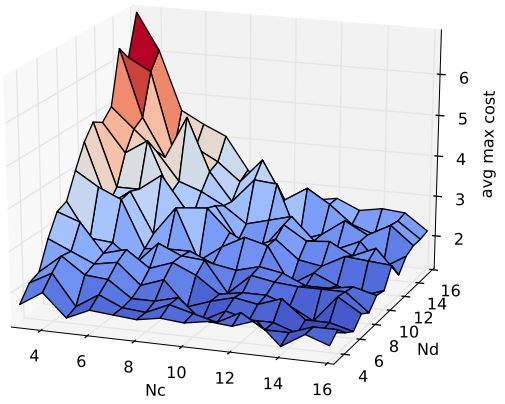

Figure 8: Average worst case individual cost over 20 simulations (`skipcount` is 10).

Using (6) as a predictor for the number of repository accesses, the cost to download N_d objects can be written as

$$\lceil N_d/N_c \rceil w_l + 2(N_d - \lceil N_d/N_c \rceil)w_s \quad (7)$$

Assuming w_s is low enough to be negligible and normalizing with the nominal cost of $N_d w_l$, the predicted normalized cost under the mechanism proposed in this paper is $\lceil N_d/N_c \rceil / N_d$, meaning that given $N_c \geq N_d$, the cost scales with $1/N_d$. As such the predicted energy savings are on par with those presented in [17], but note that those results would also be better if IEEE 802.11n had been used.

The main advantage the of feedback method in this paper is that does not require off-line optimization and therefore is able to handle uncertainties better, as show in Section 9.

9. RANDOM INITIAL STATE

Relaxing the assumption on the initial system state will give further insight into how the feedback based solver will handle a more realistic scenario. Clients might enter the system with some data objects already collected, others might join with empty caches at a later stage. Since the algorithm assumes all relevant information is part of the current system state, the exact events leading up to this point are irrelevant.

The result of a sequence of simulations, 20 for each (N_d, N_c) pair, Figure 8 shows the average max cost in a system with a random initial state, where each client possesses a random number of objects (uniformly distributed in $[0, N_d]$). The objects are in each case also picked at random, with uniform probability.

On average, a client in these simulations enters the system with half of the objects already in possession. It would therefore be reasonable to expect that the expected max cost would be lower than the $\lceil N_d/N_c \rceil$ rule would predict, but the simulations suggest otherwise. The explanation for this is that some clients finish their sets early, thereby forcing others to pay for many repository accesses, which in turn increases the max costs.

10. CONCLUSIONS

This paper has presented a proposal for how to build a peer-to-peer content distribution scheme for mobile client by providing mechanisms for enforcing cooperation agreements. The resulting barter trade like economy has been analyzed using a heuristic mechanism for finding cooperation groups. The algorithm is based on the control of a cost function, which minimization is considered to benefit the economy.

Simulations indicate that the traffic to central nodes can be reduced by a factor of $\lceil N_d/N_c \rceil / N_d$, which is beneficial both for the service provider and the clients, assuming remote traffic is more expensive in terms of energy or traffic fees.

It can also be seen that a measure of patience, that is a willingness to wait for a suitable cooperation partner rather than going for the central repository, improves the effectiveness of the exchange system, making it possible to trade cost for latency if so desired.

The system is shown to work both under assumptions of empty and random initial state, though starting out with large variances in cache contents clearly reduces the effectiveness of the algorithm.

11. FUTURE RESEARCH

Though promising even in its current state, several possibilities for extensions and refinements present themselves.

The simplistic model of data objects can easily be generalized to account for varying data object sizes. This will result in weights being introduced in (2) that will then propagate naturally into the decision algorithm through the partial derivatives.

In the scenarios presented in this paper, the service a client can provide a cooperation parter is limited to file distribution but it would be possible to trade heterogeneous services, for example file access for computations or GPS readings, using energy cost as the base for the economy.

Removing the necessity for a central mechanism is another goal to make the scheme easier to deploy in an ad-hoc manner. Taking the role of the central node, thereby performing the arbitration and distribution of encryption keys, is a service that a client could offer its peers. This extension is an important step in creating more loosely coupled and self governing systems.

The heuristic algorithm could be modified to explicitly decompose the population into groups based on the ratio of N_d and N_c. This could improve performance and computational efficiency, as well as allowing for a distributed solver.

12. ACKNOWLEDGEMENTS

This work has been partially funded by the Lund Center for Control of Complex Engineering Systems (LCCC) and the Excellence Center at Linköping - Lund on Information Technology (ELLIIT).

13. REFERENCES

[1] Matplotlib. `http://matplotlib.org`.
[2] NetworkX. `http://networkx.github.io`.
[3] Python programming language – official website. `http://python.org`.
[4] Androutsellis-Theotokis, Stephanos, and D. Spinellis. A survey of peer-to-peer content distribution

technologies. *ACM Computing Survey*, 36(4):335–371, Dec. 2004.

[5] I. Ashlagi, D. Gamarnik, M. A. Rees, and A. E. Roth. The need for (long) chains in kidney exchange. National Bureau of Economic Research. Market Design Working Group Meeting, 2011.

[6] R. Axelrod. *Evolution of cooperation*. Basic Books, 2006.

[7] G. Ding and B. Bhargava. Peer-to-peer file-sharing over mobile ad hoc networks. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004.

[8] C. D. Godsil, G. Royle, and C. Godsil. *Algebraic graph theory*, volume 8. Springer New York, 2001.

[9] D. Halperin, B. Greensteiny, A. Shethy, and D. Wetherally. Demystifying 802.11n power consumption. In *Proceedings of the 2010 Workshop on Power Aware Computing and Systems*, 2012.

[10] R. Karp. Reducibility among combinatorial problems. In R. Miller, J. Thatcher, and J. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972.

[11] G. Kortuem, J. Schneider, D. Preuitt, T. Thompson, S. Fickas, and Z. Segall. When peer-to-peer comes face-to-face: collaborative peer-to-peer computing in mobile ad-hoc networks. In *Proceedings of the First International Conference on Peer-to-Peer Computing*, pages 75–91, 2001.

[12] G. Kreitz and F. Niemela. Spotify – large scale, low latency, p2p music-on-demand streaming. In *Proceedings of the 10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, 2010.

[13] M. Lindberg. Exchange system simulation package. `https://www.control.lth.se/media/MikaelLindberg/exchangesystem.py`, 2013.

[14] D. Marin and M. Schnitzer. *Contracts in Trade and Transition: The Resurgence of Barter*, volume 1. The MIT Press, 1 edition, 2002.

[15] H. Shen, M. Joseph, M. Kumar, and S. Das. Precinct: A scheme for cooperative caching in mobile peer-to-peer systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, pages 57a–57a, 2005.

[16] O. Wolfson, B. Xu, and R. M. Tanner. Mobile peer-to-peer data dissemination with resource constraints. In *Proceedings of the International Conference on Mobile Data Management (2007)*, 2007.

[17] E. Yaacoub, L. Al-Kanj, Z. Dawy, S. Sharafeddine, F. Filali, and A. Abu-Dayya. A utility minimization approach for energy-aware cooperative content distribution with fairness constraints. *Transactions on Emerging Telecommunications Technologies*, 23(4):378–392, 2012.