The Age of Impatience: Optimal Replication Schemes for Opportunistic Networks

Joshua Reich
Department of Computer Science, Columbia U.
reich@columbia.edu

Augustin Chaintreau
Thomson
augustin.chaintreau@thomson.net

ABSTRACT

Multimedia content dissemination in mobile settings requires significant bandwidth. Centralized infrastructure is often either inadequate or overly expensive to fill the demand. Here, we study an alternative P2P content dissemination scheme for mobile devices (*e.g.,* smart-phones), which leverages local dedicated caches on these devices to opportunistically fulfill user requests. In our model, the *allocation* of content in the *global distributed cache* comprising the union of all local caches, determines the pattern of demand fulfillment. By selectively replicating local content at node meetings, the global cache can be driven towards a more efficient allocation. However, the allocation's efficiency itself is determined by a previously overlooked factor - the *impatience* of content requesters. By describing user impatience in the form of any monotonically decreasing *delay-utility* functions, we show that an optimal allocation can be efficient computed or approximated. As users become increasingly impatient, the optimal allocation varies steadily between uniform and highly-skewed towards popular content.

Moreover, in opportunistic environments, the global cache state may be difficult or impossible to obtain, requiring that replication decisions be made using only local knowledge. We develop a reactive distributed algorithm, *Query Counting Replication (QCR)* that for any delay-utility function drives the global cache towards the optimal allocation - without use of any explicit estimators or control channel information. We validate our techniques on real-world contact traces, demonstrating the robustness of our analytic results in the face of heterogeneous meeting rates and bursty contacts. We find QCR compares favorably to a variety of heuristic *perfect control-channel* competitors.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communications; C.4 [**Performance of Systems**]: Modeling Techniques

General Terms

Algorithms, Performance

Keywords

Replication, Peer-to-peer network, Content Delivery, Delay Tolerant Networks, Opportunistic Networks

1. INTRODUCTION

As smartphones capable of displaying, storing, and transmitting media content continue to proliferate, the problem of how to distribute content to these devices becomes ever more timely. A naive approach to content dissemination would leverage a centralized model in which content providers send content directly to users through centralized infrastructure. In fact, that is how practically all existing mobile content dissemination systems operate today. However, as recent events show the growth of demand for bandwidth may far outpace growth of centralized infrastructure capacity [1]. Moreover, even when centralized infrastructure is both present and adequate to meet demand, third-party providers may find use of this infrastructure too costly to serve as the primary means of fulfilling user demand.

To address this challenge, we examine an alternative *opportunistic* content dissemination schemes for mobile devices. These schemes tap the potentially vast reservoir of capacity latent in currently unused communication opportunities between the short-range radios (*e.g.,* Bluetooth, 802.11) of smartphones. Leveraging short-range communication does not come without cost or complication. Particularly, users will need to tolerate both the *energy drain* from additional short-range radio use and the *fulfillment delay* encountered by nodes forced to wait until they meet peers who have the content needed to fulfill their requests. For now we assume that energy drain will be tolerable and focus on understanding how content can be disseminated so as to minimize the impact of fulfillment delay on overall user satisfaction. Once this is better understood, future work may return to rigorously address the issue of energy efficiency.

In order to understand the impact of delayed fulfillment on user satisfaction, we develop a model predicting fulfillment delay patterns and combine this with some monotonically decreasing *delay-utility* function mapping delay to utility. We then show how the aggregate expected utility can be calculated. This enables us to investigate how this quantity may be maximized (minimizing the impact of delay on

[1]see "Customers Angered as iPhones Overload AT&T" by J. Wortham in *New York Times*, Sept. 3rd, 2009.

our users) by manipulating local cache content. Additionally, one could utilize the aggregate expected utility to determine whether opportunistic content dissemination even makes sense in a given scenario, by assessing whether it is above or below the system designer's chosen "break-even" point. Interestingly, our use of delay-utility functions enables us to answer these questions over the full spectrum of user *impatience* responses (*e.g.,* as delay increases, the user's likelihood of continuing to wait for content, decreases).

In the aforementioned model, the *allocation* of content in the *global distributed cache* comprising the union of all local caches, directly determines the pattern of demand fulfillment - and along with the delay utility function, the expected aggregate utility. By selectively replicating local content as node meetings provide the opportunity, the global cache can be driven towards a more efficient allocation.

As a motivating example, consider that an imaginary startup *VideoForU* - having already noted that users are willing to use systems that require them to donate resources which provide them with delayed content at the right price (*e.g.,* Bittorrent) - decides to provide 15 minute video shows with embedded commercial content from a catalog of 500 available episodes (this catalog changes every so often - perhaps once a week). VideoForU manages to sign up 5000 users, who agree to dedicate a 3-episode cache on their local device's memory for use by VideoForU's protocol. VideoForU can now seed one or two copies of each episode into the global cache (by using cellular infrastructure, or basestations run by VideoForU). They then let their protocol, running on the users devices, replicate content and fulfill user requests as chance meetings between users provide opportunity to do so. Assuming that the users's impatience is known (*i.e.,* the probability that a user, having waited time t, will not watch the content that she requested), via previous survey or feedback, VideoForU can design their replication protocol so as to maximize the total number of videos and embedded commercials, watched - the only question is how.

Making the answer to this question even more difficult is the fact that, for the same reasons as above (unpredictable mobility and resultant sporadic contacts), it may be difficult to gather global knowledge of the network's state. Consequently, we seek to develop distributed mechanisms capable of producing optimal or approximately optimal allocations, without needing to know the system's global state.

In this paper, we make the following contributions:

- We demonstrate that user impatience plays a critical role in determining the optimal allocation for disseminating content. We define the social welfare of a mobile P2P caching system for any delay-utility and global cache allocation (Section 3). Furthermore, we demonstrate that the optimal allocation can be computed efficiently in a centralized manner. Under the simplified assumption of homogeneous meeting rates, we show that the corresponding optimal cache allocation is known in closed form for a general class of delay-utility functions (Section 4). These results indicate that, as the user population becomes increasingly impatient, the optimal allocation changes radically: it varies steadily between a uniform allocation dividing the global cache between all content items, and a highly-skewed allocation in which popular items receive a disproportionate share of the global cache.
- Inspired by these results, we develop a reactive distributed algorithm, *Query Counting Replication (QCR)* that for any delay-utility function drives the global cache towards the optimal allocation. Moreover QCR does so without use of any explicit estimators or control channel information. Further, we show the implementation of QCR in opportunistic environment's is non-trivial and demonstrate a novel technique *Mandate Routing* to avoid potential pathologies that arise in insufficiently fluid settings (Section 5).
- Finally, we validate our techniques on real-world contact traces, demonstrating the robustness of our analytic results in the face of heterogeneous meeting rates and bursty contacts. We find QCR compares favorably to a variety of heuristic competitors, despite those competitors having access to a *perfect control-channel* and QCR relying solely on locally available information (Section 6).

2. RELATED WORK

Networks that leverage local connection opportunities to communicate in a delay tolerant manner can be classified into two categories. The first category, featuring networks such as DieselNet [1] or KioskNet [22], involves nodes with scheduled or controlled routes, and routing protocols designed to communicate with predictable latency. The second category contains network featuring unpredictable mobility [8, 4] that may be used in an opportunistic manner. In this case, it is infeasible to provide strict guarantees on message delivery time. However, opportunistic contacts may greatly enhance the performance of many peer-to-peer (P2P) applications: as proposed for website prefetching in the 7DS architecture [20], and podcast dissemination (series of content items on a channel), in the Podnet project [14]. It is into this second category that the content dissemination problem we investigate here falls. The performance of some of these systems have been analyzed from a hit-rate or delay standpoint [15, 11] for the case of a persistent demand.

Much previous work in the context of opportunistic networks has used utility functions as local states variables, both for unicast routing and publish-subscribe applications. The routing protocol PROPHET [16] uses past information to predict delivery probability. The RAPID protocol generalizes this principle into an inference algorithm which accounts for several metrics related to delay [1], while CAR [18] proposes the use of Kalman filtering to improve the prediction's accuracy. The impact of using different utility functions has been analyzed for single-copy routing schemes [24], buffer management optimization [12], and the use of error-correcting code [10]. In the context of pub-sub applications, utility functions were introduced to either predict user future demands [23], or leverage uneven distributions of demand and user proximity [2, 6]. Other advanced cache management protocols includes utilizing filters [7] and social relationships between mobile users in community [26].

In general, such use of utility functions helps a system to distinguish on-the-fly which intermediate node is the most likely to succeed (*i.e.,* for unicast routing, moving a packet closer to its destination, or, for pub-sub applications, facilitating dissemination to subscribing nodes). The performance of all these schemes are in general difficult to analyze both due to their complexity, and the interaction between local decisions using estimated utility and the global effect on network performance. Our work significantly departs from this closely-related work in two ways. The first is that instead of using (local) utility as an *intermediate* quantity used to estimate one or several parameters informing protocols, we take (global) utility as an *end-measure* for network efficiency (*i.e.,* the system's performance as it is perceived by users in aggregate). At no time during the course of the protocols is (local) utility estimated. Rather we study the effect of using light-weight replication protocols on the global utility of the network which the objective function we aim to maximize. The second difference is that we account for a general behavior of users with regard to delay, defining the global utility (or social welfare) as a function of any individually experienced delay-utilities (previous work either ignores user impatience or implicitly accounts for it using a fixed step function). A similar approach had been used for congestion control [13], and wireless scheduling [17], but not so far for content dissemination in opportunistic networks.

Replication protocols were first introduced for unstructured P2P systems deployed on wired networks, as a way to increase data availability and hence to limit search traffic [5, 25]. Assuming that nodes search for files in random peers, it was shown [5] that for each fulfilled request, creating replicas in the set of nodes used for the search (*i.e., path-replication*) achieves a square root allocation: a file i requested with probability p_i has a number of replicas proportional to $\sqrt{p_i}$ at equilibrium. This allocation was shown to lead to an optimal number of messages overall exchanged in the system. Assuming that nodes use an expanding ring search, an allocation where each file is replicated in proportion of its probability p_i was shown to be optimal [25]. The meeting between unpredictable mobile nodes can in some sense be compared to a random search, and we extend the results above for a P2P system deployed on top of opportunistic contacts between mobile devices. The main novelty is that the behavior of user with regard to delay greatly impacts which algorithms to select for optimal performance.

Immediately before completing the final version of this paper, we heard of an on-going effort to characterize a related channel selection problem [9]. The algorithm proposed in this case uses an estimate of dissemination time and a Metropolis-Hasting adaptive scheme. One difference between the two approaches is that we show, because the optimal allocation satisfies a simple balance condition, that even simple algorithms which do no maintain any estimates of dissemination time or current cache allocation are optimal for a known delay-utility function. Another difference is that we also prove that the submodularity property for the cache allocation can be established even when contacts and delay-utility functions are not homogeneous.

3. EFFICIENCY OF P2P CACHING

Some nodes store content which they use to fulfill requests of the nodes they meet. In this section, we assume that the allocation of content to these nodes is fixed. We show that the global efficiency of such a system can be measured with an objective function parameterized by a delay-utility function representing the average user's impatience behavior.

3.1 Node Types, Content Cache

Each node in the P2P system may be a *client*, a *server*, or both. The set of client nodes is denoted by $\mathcal{C}$, we generally denote its size by N. Each client demands and consumes content as described in Section 3.3. The set of all server nodes is denoted by $\mathcal{S}$. Servers maintain a cache in order to make it available to interested clients (when such clients are met). This includes in particular the two following scenarios:

Dedicated nodes server and client populations are separate (*i.e.,* $\mathcal{C} \cap \mathcal{S} = \emptyset$).

Pure P2P all nodes act as both server and client (*i.e.,* $\mathcal{C} = \mathcal{S}$).

The dedicated node case resembles a managed P2P system, where delivery of content is assisted by special types of nodes (*e.g.,* buses or throwboxes [1], kiosks [22]). The pure P2P case denotes a cooperative setting where all nodes (*e.g.,* users's cell-phones [20, 14]) request content as well as help deliver content to others. The motivating scenario, mentioned in the introduction, of VideoForU is likely to resemble the Pure P2P scenario, especially if as little content as possible is seeded with cellular infrastructure.

Caches in Server nodes.

The main variable of interest in the system is the cache content across all server nodes. In this section we assume it to be fixed; in practice the global cache dynamically evolves through a replication protocol (see section 5).

For any item i and m in $\mathcal{S}$, we define $x_{i,m}$ to be one if server node m possesses a copy of item i, and zero otherwise. The matrix $\mathbf{x} = (x_{i,m})_{i\in I, m\in\mathcal{S}}$ represents the state of the global distributed cache. We denote the total number of replicas of item i present in the system by $x_i = \sum_{m\in\mathcal{S}} x_{i,m}$.

In the rest of this paper, we assume that all servers have the same cache size so that they can contain up to ρ content items (all items are assumed to have the same size). This is not a critical assumption and most of the following results can be extended to caches or content items of differing sizes. It follows that a content allocation $\mathbf{x}$ in server nodes is feasible if and only if:

$$\forall m \in \mathcal{S}\,,\ \sum_{i\in I} x_{i,m} \le \rho\,.$$

3.2 Representing Impatience as Delay-utility

In contrast with previous work in P2P networks, P2P content dissemination over an opportunistic mobile network induces a non-negligible *fulfillment delay* between the time a request is made by a client node and the time that it is fulfilled. This delay depends on the current cache allocation, as a request is fulfilled the next time the requesting node meets another node possessing a copy of the desired content. The term *impatience* refers to the phenomenon that users become decreasingly satisfied (or increasingly dissatisfied) with the delays they experience. A *delay-utility* function $h(t)$ can be used to characterize this phenomenon of user impatience in analytic terms, where the value of this function is monotonically decreasing with time (as increasing delay will not translate into increasing satisfaction).

Since different types of content may be subject to differing user expectations, we allow each content item i in the set of all system-wide content items available I, its own delay-utility function h_i. The value $h_i(t)$ denotes the gain for the network resulting from delayed fulfillment of a request for item i when this occurs t time units after the request was created. This value can be negative, which denotes that this delayed fulfillment generates a disutility, or a cost for the network. Note that t is related here to the user's waiting time, not to the time elapsed since the creation of the item. Currently, we decided to use the same set of delay-utility functions for all users. One can therefore interpret $h_i(t)$ as the average among users of the gain produced when a request is fulfilled after waiting for t time units. All the results we present generalize to users following different functions, but we choose to follow a simple average function to avoid notational issues, and to keep the system design simple.

We now present several examples of delay-utility functions corresponding to different perceptions of the performance of a P2P caching system by the users.

Advertising Revenue.

Assuming content items are videos starting with embedded advertisements, and that the network provider receives a constant unit revenue each time a commercial is watched by a user (a potential business plan for the scenario of VideoForU). In this case, the delay-utility function simply denotes the probability that a user watches a given video when she receives the content t time after it was requested. Two possible function families modeling this situation are:

Step function $h_\tau^{(\mathrm{s})} : t \mapsto \mathbb{I}_{\{t \le \tau\}}$.

Exponential function $h_\nu^{(\mathrm{e})} : t \mapsto \exp(-\nu t)$.

The former models a case where all users stop being interested in seeing the item after waiting for the same amount of time. In the second case, the population of users is more mixed: at any time, a given fraction of users is susceptible to losing interest in the content.

Time-Critical Information.

Assuming the content exchanged by nodes deals with an emergency, or a classified advertisement for a highly demanded and rare product (*i.e.,* a well located apartment). In such cases, as opposed to the previous model the value of receiving this piece can start from a high value but very quickly diminish. It is possible to capture such a behavior by a delay-utility presenting a large reward for a prompt demand fulfillment.

Inverse power $h_\alpha^{(\mathrm{p})} : t \mapsto \dfrac{t^{1-\alpha}}{\alpha - 1}$. with $\alpha > 1$

Note that the value of delivering an item immediately in this case is arbitrarily large ($h(0^+) = \infty$). Such immediate delivery can occur when a node is both a server and a user, as the local cache may already contain the item requested. To exclude this case, we restrict the use of such delay-utility functions to the Dedicated node case.

Waiting Cost.

In some situations, such as a patch needed to use or update a particular application, users may request for an item and insist on receiving it no matter how long it takes, becoming with time increasing upset because of tardy fulfillment. As an example, the time a user spent with an outdated version of a software application may be related with the risk of being infected by a new virus, and hence incurring a high cost. One can consider to represent such cases a delay-utility function that grows increasingly more negative with time, corresponding to a cost for the user and the network.

Negative power $h_\alpha^{(\mathrm{p})}$ as above with $\alpha < 1$

Negative Logarithm $h_1^{(\mathrm{p})} : t \mapsto -\ln(t)$.

The negative logarithm corresponds to the limit as α approaches 1. It features both a high value for fast fulfillment of request and a negative cost becoming unbounded as waiting time grows.

We plot on Figure 1 illustration of delay-utility functions for the three motivating examples presented above.

To simplify the presentation below, we will assume in this paper that h admits a finite limit at time $t = 0$, (*i.e.,* $h(0^+) < \infty$). This excludes the inverse power and the negative logarithm delay-utility functions introduced above. These functions can be considered in the dedicated node case where the exact same results hold, as shown in [21].

3.3 Client Demand

Clients register their demand for content in the form of *requests.* As in previous work, we assume that the process of demand for different items follows different rates, reflecting differing content popularity. We denote by d_i the total rate of demand for item i. In the rest of this paper, we assume any arbitrary values of d_i. As an example of demand distribution, one may use

Pareto with parameter $\omega > 0$: $d_i \propto i^{-\omega}$ for all $i \in I$.

In simulation we use a Pareto popularity distribution, generally considered as representative of content popularity.

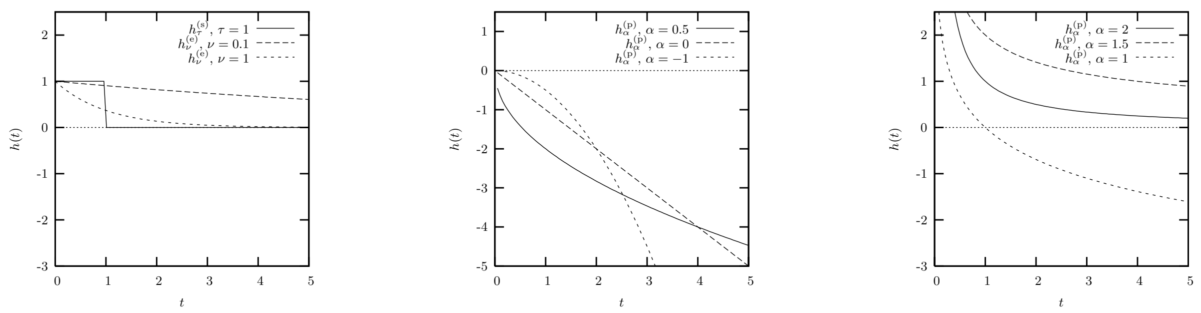
We denote by $\pi_{i,n}$ the relative likeliness of a demand for item i arising at node n, where $\sum_{n \in \mathcal{C}} \pi_{i,n} = 1$. In other words, node n creates a new request for item i with a rate equal to $d_i \pi_{i,n}$. One can generally assume that different populations of nodes have different popularity profile, generally captured in the values of $\pi_{i,n}$. Otherwise, we can assume that items, especially the ones with the highest demand, are popular equally among all network nodes. This corresponds to the case where $\pi_{i,n} = 1/|\mathcal{C}|$.

3.4 Node Mobility

As all nodes (whether client or server) move in a given area, they occasionally meet other nodes - these meetings provide the opportunity for replication of cache content and fulfillment of outstanding requests. For simplicity and as a way to compare different P2P caching schemes, we focus on a case where contacts between clients and server nodes follow independent and memory less processes. In other words, we neglect the time dependence and correlation between meeting times of different pairs which may arise due to complex properties of mobility. In that case the process of contacts between two nodes m and n is entirely characterized by their contact intensity (the number of contacts between them per unit of time), which we denote by $\mu_{m,n}$.

Our model can be defined for any contact processes, this is what we simulate in Section 6 for a comparison using real traces. The memoryless assumption helps us to understand what are optimal strategies in a simple case before evaluating them using real traces for a complete validation of these trends. Two contact models can be considered:

Discrete time The system evolves in a synchronous manner, in a sequence of time slots with duration δ. For each time slot, we assume node contacts occur independently with probability $\mu_{m,n} \cdot \delta$ (for $m \in \mathcal{S}$, $n \in \mathcal{C}$).



(a) Advertising Revenue (b) Time-Critical Information (c) Waiting Cost

Figure 1: Delay-utility functions used for advertising revenue (left), time-critical information (middle) and waiting cost (right).

Continuous time The system evolves in an asynchronous manner, so that events may occur in continuous time. We assume that node contacts occur according to a Poisson Process with rate $\mu_{m,n}$ (for $m \in \mathcal{S}$, $n \in \mathcal{C}$).

Note that when δ is small compared to any other time in the system, the discrete time model approaches the continuous time model. In this paper, whenever space permits we write results for both contact model, focusing on the continuous case. Simulations results, which are based on discrete event processes, confirm the good match between our continuous time analysis and the discrete time dynamics of a real system.

The system is said to follow *homogeneous contacts* if we have $\mu_{m,n} = \mu$ for all nodes $m \in \mathcal{S}$ and $n \in \mathcal{C}$. This case corresponds to a population of nodes with similar characteristics where all meeting are equally likely, as for instance it may be between the participants of a special event.

3.5 Content allocation objective

Demand arises in our P2P system according to content popularity, and is served as a function of mobility and content availability, captured through variables $\mathbf{x} = (x_{i,m})_{i\in I, m\in\mathcal{S}}$.

We define $U_{i,n}(\mathbf{x})$ to be the expected gain generated by a request for item i created by client node n. Following our model of users's impatience, this expected gain is equal to $\mathbb{E}[h_i(Y)]$ where Y denotes the time needed to fulfill this request, which itself critically depends on the availability of item i in servers's caches.

The total utility perceived by all clients in the system, also called *social welfare*, may then be written as:

$$U(\mathbf{x}) = \sum_{i\in I} d_i \sum_{n\in\mathcal{C}} \pi_{i,n} U_{i,n}(\mathbf{x})\,. \qquad (1)$$

A good allocation $\mathbf{x}$ of content across the global cache is one that results in a high social welfare. Note that this objective combines the effects of delay on the gains perceived by users, the popularity of files, as well as the cache allocation.

In the remaining of this section, we derive an expression for $U_{i,n}(\mathbf{x})$, based on the *differential delay-utility function*, which will be instrumental in deriving some of its properties.

Differential delay-utility function.

We denote this function by c_i for the continuous time contact model (resp. Δc_i for the discrete time contact model). These functions are simply defined by

$$c_i(t) = -\frac{dh_i}{dt}(t)\,,\text{ and } \Delta c_i(k\delta) = h_i(k\delta) - h_i\left((k+1)\,\delta\right)\,.$$

The values of $c_i(t)$ and $\Delta c_i(k\delta)$ are always positive as h_i is a non-increasing function. The value of c_i (resp. Δc_i) represents the additional loss of utility, which is incurred per additional unit of time spent waiting (resp. the loss of utility incurred for waiting an additional time slot).

We present in the second line of Table 1 the expression for c_i for all the delay-utility functions introduced above. Note that when h_i is not differentiable (like for the step function), it may happen that c_i is not defined as a function but as the derivative measure in the sense of the distribution.

General expression for $U_{i,n}(\mathbf{x})$.

Following a slight abuse of notation, we set by convention $x_{i,n} = 0$ when n is not a server node (*i.e.,* $n \notin \mathcal{S}$). With this notation, we find the following expressions for $U_{i,n}$.

Lemma 1. *In the discrete time contact model, $U_{i,n}(\mathbf{x})$ is*

$$h_i(\delta) - (1 - x_{i,n}) \sum_{k\geq 1} \prod_{m\in\mathcal{S}} \left(1 - x_{i,m}\mu_{m,n}\delta\right)^k c_i(k\cdot\delta)\,,$$

For the continuous time contact model, $U_{i,n}(\mathbf{x})$ is

$$h_i(0^+) - (1 - x_{i,n}) \int_0^\infty \exp\left(-t \sum_{m\in\mathcal{S}} x_{i,m}\mu_{m,n}\right) c_i(t)dt\,.$$

The proof follows from the memory less property of contacts and the expectation as obtained in integration by part:

$$\mathbb{E}[h(Y)] = h(0^+) + \int_0^\infty (1 - F_Y(t))h'(t)dh\,.$$

The term $(1 - x_{i,n})$ deals with possible immediate fulfillment (*i.e.,* request created by a node that already contains this item in its local cache). For more details, see [21].

Homogeneous contact case.

If we assume homogeneous contacts (*i.e.,* $\mu_{m,n} = \mu$), the general expressions above simplifies. In particular, the utility depends on $(x_{i,n})_{i\in I, n\in\mathcal{S}}$ only via the number of copies present in the system for each item $(x_i)_{i\in I}$.

First, in the dedicated node case (*i.e.,* $\mathcal{S} \cap \mathcal{C} = \emptyset$), we have, respectively for the discrete time contact model and the continuous time contact model:

$$U(\mathbf{x}) = \sum_{i\in I} d_i \left(h(\delta) - \sum_{k\geq 1} (1-\mu\delta)^{x_i k}\, c_i(k\cdot\delta) \right) . \quad (2)$$

$$U(\mathbf{x}) = \sum_{i\in I} d_i \left(h(0^+) - \int_0^\infty e^{-t\mu x_i} c_i(t)dt \right) . \quad (3)$$

Similarly, for the pure P2P case, if we further assume that all $N = |\mathcal{C}| = |\mathcal{S}|$ nodes follow the same item popularity profile (*i.e.,* $\pi_{i,n} = 1/N$), we have for the two different models of contact process:

$$U(\mathbf{x}) = \sum_{i\in I} d_i \left(h(\delta) - \left(1 - \frac{x_i}{N}\right) \sum_{k\geq 1} (1-\mu\delta)^{x_i k}\, c_i(k\cdot\delta) \right) . \quad (4)$$

$$U(\mathbf{x}) = \sum_{i\in I} d_i \left(h(0^+) - \left(1 - \frac{x_i}{N}\right) \int_0^\infty e^{-t\mu x_i} c_i(t)dt \right) . \quad (5)$$

All these expressions follows from a simple application of Lemma 1 (see [21] for complete details).

4. OPTIMAL CACHE ALLOCATION

The *social welfare* defined above measures the efficiency of cache allocation which captures users's requests and impatience behavior. Finding the best cache allocation is then equivalent to solving the following optimization problem:

$$\max \left\{ U(\mathbf{x}) \;\middle|\; x_{i,n} \in \{0,1\}, \forall n \in \mathcal{S}, \sum_{i\in I} x_{i,n} \leq \rho \right\} . \quad (6)$$

4.1 Submodularity, Centralized computation

A function f that maps subset of $\mathcal{S}$ to a real number is said to be *sub-modular* if it satisfies the following property: $\forall A \subseteq B \subseteq \mathcal{S}$, $\forall m \in \mathcal{S}$, $f(A\cup\{m\}) - f(A) \geq f(B\cup\{m\}) - f(B)$.

This property generalizes to set functions the concavity property defined for continuous variables. Colloquially this is referred to as "diminishing returns" since the relative increase obtained when including new elements diminishes as the set grows.

The function $U_{i,n}(\mathbf{x})$ can be interpreted as a function that maps subset of $\mathcal{S}$ (*i.e.,* the subset of servers that possess a replica for item i) to a real number (the expected value of a request for item i created in client n). Similarly, U may be seen as a function that maps subset in $\mathcal{S} \times I$ (subsets denoting which servers possess which replica), to a real value (the social welfare). We then have the following result.

THEOREM 1. *For any item i and node n, $U_{i,n}$ is submodular. As a consequence U is submodular.*

This result can be interpreted intuitively. On the one hand, in order to increase the value of $U_{i,n}$, creating a new copy of item i (*i.e.,* including a new element in the set of servers containing a copy of i) always reduce delays and hence increases utility. On the other hand the *relative* improvement obtained when creating this copy depends on the number of copies of i already present, and it diminishes as that item is more frequently found. What is perhaps less obvious is that this result holds for any mixed client/server population of nodes, heterogeneous contact processes, and any arbitrary popularity profile.

An interesting consequence is that one can deduce from submodularity, under some additional conditions, that a greedy procedure builds a $(1-1/e)$-approximation of the maximum social welfare for given capacity constraints (see [19]). A greedy algorithm is used in Section 6 to find a cache allocation for heterogeneous contact traces.

The proof of this result uses the general expression for $U_{i,n}$ found in Lemma 1 and a few observations: First, that the expression inside the integral multiplying the differential delay-utility function is a supermodular non-increasing and non-negative function of the set of servers containing i. Second, that since the differential delay-utility function is positive, all these properties apply to the integral itself. Finally, that the product with $(1 - x_{i,n})$ preserves the supermodular non-increasing and non-negative properties. A complete formal proof can be found in [21].

In the case of homogeneous contact rates, we can obtain an even stronger result, as the social welfare only depends on the number of replicas for each item, and not on the actual subset of nodes that possess that item.

THEOREM 2. *In the homogeneous contact case, $U(\mathbf{x})$ is a concave function of $\{ x_i \mid i \in I \}$.*

The optimal values of $\{ x_i \in \{0, 1, \cdots, |\mathcal{S}|\} \mid i \in I \}$ are found by a greedy algorithm using at most $O(|I|+\rho|\mathcal{S}|\ln(|I|))$ computation steps.

Moreover, the solution of the relaxed *social welfare maximization (*i.e., *maximum value of $U(\mathbf{x})$ when $(x_i)_{i\in I}$ are allowed to take real value) can be found by gradient descent algorithm.*

The concavity property is here not surprising, as it corresponds to submodularity when the function is defined using continuous variables rather than a set. Formally, the arguments used to prove this result are quite similar to the previous proof: one leverages previous expressions which feature the product with the differential delay-utility function, and then use the fact that the family of convex non-negative non-increasing functions is closed under product.

The greedy algorithm follows a simple operation repeated once for each copy that can be cached ($\rho|\mathcal{S}|$ steps in total): at each time step from the current cache allocation, it adds a copy for the item that brings the most significant relative increase in utility (assuming there does not exist already $|\mathcal{S}|$ copies of this item). By doing so, the algorithm is likely to select first popular items. As the popular items fill the cache with copies, the relative improvement obtained for each additional copy diminishes, and the greedy rule will choose to create copies for other less popular items. The diminishing return property ensures that this greedy algorithm selects the optimal cache allocation. For the same reason, starting from a cache allocation, a hill climbing algorithm with full knowledge can reach the optimal cache allocation only from local manipulation of cache between nodes that are currently meeting. A formal proof of these results can be found in [21].

4.2 Characterizing the optimal allocation

In the homogeneous contact case, whenever x_i only takes integer values, it can be difficult to grasp a simple expression for the allocation maximizing social welfare, as it is subject to boundary and rounding effect. However, when the number of servers is large, x_i may take larger values, in particular for popular items. Hence, the difference between the optimal allocation and the solution of the *relaxed* optimization (where $(x_i)_{i\in I}$ may take real values, as defined in Theorem 2) tends to become small. The latter is then a good approximation of the former. In addition, when the number of clients N becomes large, the difference between the dedicated node case and the pure P2P case tends to become negligible, as the correcting terms $(1-\frac{x_i}{N})$ in Eq.(4) and (5) approaches 1.

We show in this section that the solution of the relaxed optimization problem satisfies a simple equilibrium conditions. Although we only derive this condition in the continuous time contact model, a similar condition can be found in the discrete case model.

PROPERTY 1. *We consider the continuous time contact and dedicated node case. Let $\tilde{x}$ be the solution of the relaxed social welfare maximization (as defined in Theorem 2), then*

$$\forall i,j\,,\ \tilde{x}_i=|\mathcal{S}|\ \text{ or }\ \tilde{x}_j=|\mathcal{S}|\ \text{ or }\ d_i\cdot\varphi(\tilde{x}_i)=d_j\cdot\varphi(\tilde{x}_j)\,.$$

where we define φ as $\varphi : x \mapsto \int_0^\infty \mu t e^{-\mu t x} c(t)dt\,.$

This property states that, at the optimal solution of the relaxed cache allocation problem, the amount of copies created for all items depends on their popularity exactly in the same way: via a unique function φ defined *independently* of i. This equality holds only when the number of copies is not limited by the number of servers, otherwise it becomes an inequality.

This property follows from a simple derivation of the social welfare, as $\frac{\partial U}{\partial x_i}(\mathbf{x}) = d_i\varphi(x_i)$, which may be deduced from Eq.(3). At the optimal solution of the relaxed allocation problem, these derivatives should all be equal except for the value of x_i that are on the boundary of the domain (*i.e.,* when $x_i=|\mathcal{S}|$). If two points are in the interior and the derivative above differ, it is possible to modify $\tilde{x}$ slightly to remain under the capacity constraint and obtain an even larger social welfare, which would be absurd.

The function φ can always be defined a transform of the delay-utility function. For different choices of delay-utility, it leads to simple expressions which can be found in Table 1. As an example, when all items exhibit power delay-utility ($h_i = h_\alpha^{(\mathrm{p})}$), φ is a power function as well. The property implies then that, for all item i that are within the boundary conditions (*i.e.,* $x_i < |\mathcal{S}|$), the product $(\tilde{x}_i)^{2-\alpha}d_i$ is a constant that does not depend on i. We deduce that the optimal cache allocation for the relaxed problem resembles the distribution where $x_i \propto d_i^{1/(2-\alpha)}$, as shown in Figure 2.

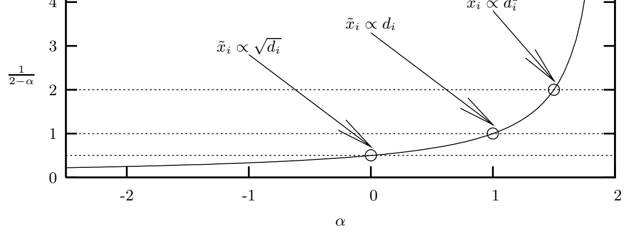


Figure 2: Coefficient of the optimal allocation for power delay-utility functions, as a function of α.

Note that, as α approaches 1 (*i.e.,* delay-utility is a negative logarithm), the social welfare is maximized when each item receives a fraction of cache proportional to its demand. For smaller values of α (*i.e.,* waiting time cost) the optimal cache allocation becomes more egalitarian, it tends to uniform as α becomes arbitrary small. For larger values of α (*i.e.,* time-critical information), the optimal allocation becomes more and more skewed towards popular items (which are likely to give the best reward); as α approaches 2, the most demanded items completely dominate the cache. Similar qualitative observations hold for the step-function utility, the optimal allocation is more complex but again varies between these two extreme cases.

5. DISTRIBUTED OPTIMAL SCHEMES

The previous section establishes that the cache allocation problem admits an optimal operating point, which may in some cases be known in closed form, and can always be either computed directly or approximated in a centralized manner. When a highly available control channel is available, using such a centralized approach is feasible. However, making each local decisions based on global information maintained using this control channel seems to reach the optimal allocation only at a prohibitive cost.

In this section, we demonstrate that one does not need to maintain global information, or know the demand of items *a priori*, to approach the optimal cache allocation. This results in a drastic reduction of overhead and makes such caching service *possible* where no infrastructure is available.

We show that a simple reactive protocol, generalizing replication techniques introduced in the P2P literature, are able to approach the optimal allocation using only local information. In order to build a low-overhead reactive protocol for the opportunistic setting, two particular challenges need be overcome:

- We must understand how to construct a replication strategy that reacts naturally to the demand for and availability of content (Section 5.1), while also properly adapting our replication strategy to impatience of users (Section 5.2). A successful strategy will allow us to approach the optimal efficiency when the system reaches equilibrium.

- We must ensure that the replication technique is implemented in such a way that ensures the convergence towards the equilibrium. This challenge proves to be non-trivial in the opportunistic context for the strategies we examine (Section 5.3).

5.1 Query Counting Replication

We propose a general class of distributed schemes, that we call *Query Counting Replication (QCR)*. QCR *implicitly adapts* to the current allocation of data and collection of requests, without storing or sharing explicit estimators. QCR achieves this by keeping a *query count* for each new request made by the node. Whenever a request is fulfilled for a particular item, the final value of the query counter is used to regulate the number of new replicas made of that item. The function ψ that maps the value of the query counter to the amount of replicas produced is called the *reaction* function. We describe in Section 5.2 precisely how it should be set, given knowledge of user impatience.

As an example, consider a VideoForU client *Amy* who begins requesting a copy of video i. Each time Amy (or more precisely Amy's phone) subsequently meets another VideoForU node, Amy's phone queries the node met for a copy of item i and increments the query counter associated with i. If after nine meetings Amy's finally meets a node possessing a copy of item i and receives a copy of video i, according to the above rule, Amy's phone will create $\psi(9)$ replicas of this item and transmit them proactively to other nodes storing VideoForU content when the opportunity arise. This principle generalizes path replication [5] where $\psi(y)$ was a linear function of y.

Contacts between mobile nodes are unpredictable, hence, as Amy's phone distributes replicas, it may encounter nodes that already contain this item. We then distinguish two implementations: replication *without rewriting* where such contact is simply ignored, or replication *with rewriting* where such contacts decreases by one the number of replica to be distributed, even though no new copy can be made.

5.2 Tuning replication for optimal allocation

We now describe how to choose the reaction function ψ depending on users's impatience. We first observe that the expected value of the query counter for different item i is proportional to $|\mathcal{S}|/x_i$, since whenever a node is met there is roughly a probability $x_i/|\mathcal{S}|$ that it contains item i in its cache. Hence, we can assume as a first order approximation that approximately $\psi(|\mathcal{S}|/x_i)$ replicas are made for each request of that items. Inversely, as a consequence of random replacement in cache, each new replicas being produced for any items erases a replica for item i with probability $x_i/(\rho|\mathcal{S}|)$. When *rewriting* is allowed, one should account for all replicas created (including the one created for the same item), we focus on this case for the analysis. Otherwise one should consider all replicas created for all other items. As a consequence, the number of copies for each item follows the system of differential equations:

$$\forall i \in I\,,\ \frac{dx_i}{dt} = d_i \cdot \psi(\frac{|\mathcal{S}|}{x_i}) - \frac{x_i}{\rho|\mathcal{S}|} \cdot \sum_{j\in I} d_j \psi(\frac{|\mathcal{S}|}{x_j})\,. \qquad (7)$$

Assuming the system converges to a stable steady state, the creation of copies should compensate exactly for their deletion by replacement. In other words a stable solution of this equation satisfies

$$\forall i \in I\,,\ d_i \frac{1}{x_i} \cdot \psi(\frac{|\mathcal{S}|}{x_i}) = \frac{1}{\rho|\mathcal{S}|} \cdot \sum_{j\in I} d_j \psi(\frac{|\mathcal{S}|}{x_j})\,.$$

Note that the RHS is a constant that does not depend on i anymore, so that this implies

$$\forall i,j \in I\,,\ d_i \frac{1}{x_i} \cdot \psi(\frac{|\mathcal{S}|}{x_i}) = d_j \frac{1}{x_j} \cdot \psi(\frac{|\mathcal{S}|}{x_j})\,.$$

In other words, the steady state of this algorithm satisfies the equilibrium condition of Property 1 if and only if we have: $\forall x > 0\,,\ \frac{1}{x}\psi(\frac{|\mathcal{S}|}{x}) = \varphi(x)$ where φ is defined as in Property 1. Equivalently, $\forall y > 0\,,\ \psi(y) = \frac{|\mathcal{S}|}{y}\varphi(\frac{|\mathcal{S}|}{y})\,.$

PROPERTY 2. *The steady state of QCR satisfies the equilibrium condition of Property 1 if and only if*

$$\psi(y) \propto |\mathcal{S}|/y \int_0^\infty \mu t e^{-\mu\frac{t|\mathcal{S}|}{y}} c(t)dt\,.$$

The upshot of this result is that as long as the delay-utility function representing user impatience is known, we can always determine the number of copies QCR must make to drive the allocation towards its optimal. In particular, the optimal reaction function can be derived in a simple expression for all the delay utilities previously introduced, as seen in Table 1. This table was computed for the continuous time and dedicated node case. A similar table can be derived for the pure P2P case (see [21]). It is approximately equivalent to this one whenever the number of client nodes N is large.

5.3 Mandate routing

Up to this point we have worked under the assumption that copies can be made more or less immediately, as in classical wired P2P networks. However, in an opportunistic context this is far from true. Particularly:

- Copies can only be made when another node is met, which happens only sporadically. Creating a replica may also takes additional time. For example, when rewriting is not allowed and the node met may already have a replica of that item.
- Since cache slots are overwritten randomly, it could be that, when a replica of the item needs to be produced, this item is no longer in the possession of the node desiring to replicate it.

Mandates & Pathologies.

Because replicas cannot be simply generated immediately, QCR mechanism deployed in an opportunistic context must inherently make (either implicitly or explicitly) a set of instructions for *future* replication of item i (*i.e.,* instructions to be used later, when the possibility for execution arises). We call such an instruction a *replication mandate* or *mandate* for short.

When a meeting occurs the mandate attempts to execute itself, but as we have already discussed, the circumstances may often not allow for its execution. This dependence of mandate execution on the state of the distributed cache may throw a monkey wrench in the dynamics outlined in Section 5.2 - for if the cache deviates to much from its expected state, the rates at which a given replica population evolves may be higher or lower than expected as well. As an example, if there are many fewer than expected copies of item i in the cache, and item i was erased by later random replacement, item i may rarely be present again, so that mandates may not be executed soon in the future. An item i that, in contrast, is more frequently found, will execute its mandate more quickly and hence continue to dominate. Consequently, if mandates are simply left at their node of origin the allocation produced by any given run of QCR can diverge significantly from the target allocation, resulting in a loss of social welfare.

Our solution.

In order to address the above pathology, we need to ensure that the number of replication actions taken for each message is proportionally the same as the number of mandates produced for that message. This could be done in several ways, which all boil down to one of the following two approaches: (1) Move replicas to nodes with mandates for those replicas, or (2) Move mandates to nodes possessing the replicas which those mandates need in order to execute.

Model	Step function	Exponential decay	Inv. Power ($\alpha < 1$)	Neg. Power ($1 < \alpha < 2$)	Neg. logarithm ($\alpha = 1$)
Impatience $h(t)$	$\mathbb{I}_{\{t \le \tau\}}$	$\exp(-\nu t)$	$\dfrac{t^{1-\alpha}}{\alpha - 1}$		$-\ln(t).$
Diff. Impat. c	Dirac at $t = \tau$	density $t \mapsto \nu \exp(-\nu t)$	density $t \mapsto t^{-\alpha}$		
Gain $U(\mathbf{x})$	$\sum_i d_i(1 - e^{-\mu\tau x_i})$	$\sum_i d_i(1 - \frac{1}{1+\frac{\mu}{\nu}x_i})$	$\sum_i d_i x_i^{\alpha-1} \frac{\mu^{\alpha-1}\Gamma(2-\alpha)}{\alpha-1}$		$\sum_i d_i \ln(x_i) - \texttt{cst}$
Cond. φ (Prop 1)	$d_i \cdot \mu\tau e^{-\mu\tau x_i}$	$d_i \cdot \frac{\mu}{\nu}\left(1 + \frac{\mu}{\nu}x_i\right)^{-2}$	$d_i \cdot x_i^{\alpha-2}\mu^{\alpha-1}\Gamma(2-\alpha)$		
Reaction ψ (Prop 2)	$(\mu\tau\lvert\mathcal{S}\rvert/y)\, e^{-\frac{\mu\tau\lvert\mathcal{S}\rvert}{y}}$	$\left(2 + \frac{\nu}{\mu\lvert\mathcal{S}\rvert}y + \frac{\mu\lvert\mathcal{S}\rvert}{\nu}\frac{1}{y}\right)^{-1}$	$y^{1-\alpha}\left(\mu^{\alpha-1}\lvert\mathcal{S}\rvert^{\alpha-1}\Gamma(2-\alpha)\right)$		

Table 1: Several delay-utility functions with their associated equilibrium and reaction functions.

The former approach (*e.g.,* protecting items with current mandates from being erased by random replacement) violates the dynamics we are trying to protect and introduces significant implementation-level complexity - as we must now either replicate or protect against deletion particular messages based on locally existing mandates. While in practice these effects may be more or less severe, the second option of moving mandates to nodes with replicas provides us with a way of solving the problem *that involves no addition biasing of the overwrites, nor requires any adjustment to the mechanism of cache adjustment.* Additionally mandates are by nature quite small pieces of data, so moving them introduces little additional overhead in terms of communication and storage.

The mandate routing scheme used for the experiments shown in Section 6 is simple but can have significant impact as will be seen later. We assume that when two nodes meet, mandates are transfered with preference to the nodes possessing copies of the messages to be replicated. This ensures that most of the mandates that cannot be executed are soon transferred to appropriate nodes. Otherwise mandates are simply spread around - split evenly between the nodes. We demonstrate empirically that this simple modifications avoids divergence of QCR and is sufficient to converge towards an optimal point.

6. VALIDATION

We now conduct an empirical study of different replications algorithms in a homogeneous contact setting, as well as several traces in various mobility scenarios. The goal of this study is threefold. Firstly, to validate empirically that the rationale behind our distributed scheme does actually converge close to the optimal value we predict. Secondly, to observe quantitatively its improvement over simple heuristics. Thirdly, to test if the same scheme adapts well to contact heterogeneity present in real-world mobility traces, as well as complex time statistics and dependencies between contacts present in these.

6.1 Simulation settings

We have built a custom discrete-event, discrete-time simulator in C++ which given any input contact trace simulates demand arrival and the interactions of node meetings.

Data plots present below are the average of 15 or more trials with confidence interval corresponding to 5% and 95% percentiles. As said in Section 3.3 items are requested following Pareto distribution, here with parameter $\omega = 1$. By default we assume $\rho = 5$. Other values of ω and ρ can be found in [21]. We do not consider the additional complexity of meeting durations. Instead we work on the premise that meetings are sufficiently long for nodes to complete the protocol exchange.

Implementation of QCR.

When two nodes meet they first exchange meta data. If either nodes have outstanding requests for messages to be found in the other's cache, then each of those requests is fulfilled. For each fulfillment a gain is recorded by the simulator, based on the age of this request and the delay-utility function. Nodes maintain query counters and makes a set of new mandates for each message fulfilled (as specified in Section 5.2). After fulfillment, the nodes then execute and route all of their eligible mandates (by sharing it equally if both nodes still possess a copy of the items, otherwise give it to the only node with a copy of this item). Rewriting of copy is not allowed, which means that contacts with a node already containing a copy of this item are simply ignored.

Each item i has one *sticky* replica which cannot be erased. This implementation detail has the effect of ensuring that we do not enter an absorbing state in which certain messages have been lost through discrete stochastic effects. We include them in mandate routing as preferred nodes (they will receive 2/3 of all mandates for this particular item whenever they meet a copy with this item, or all of them if the item has been erased on this node). We believe it is a reasonable assumption for a fielded system, given that the initial seeder of a content item will likely keep that item permanently.

Competitor Algorithms.

We compare the performance of QCR against several heuristics using *perfect* control-channel information and the ability to set the cache *precisely and without restriction* to their desired allocation: **OPT** an approximation of the optimal obtained by a greedy algorithm optimizing Pb.(6). It is exactly optimal in the homogeneous case and approximately so in the heterogenous ones; **UNI**: memory is evenly allocated amongst all items; **SQRT**: memory allocation proportionally to the square root of the demand; **PROP**: memory allocation proportional to demand; **DOM**: all nodes contain the ρ most popular items.

6.2 Homogeneous contacts

We simulate a network of 50 nodes with 50 content items ($I = 50$), meeting according to a rate $\mu = 0.05$ (the absolute value of μ plays no role in the comparison between different replication algorithm). As we wish to validate our analysis is not a mere artifact of the constraints used to generate it, we focus on the pure P2P case ($|C| = |S| = N = 50$), which is the furthest from the analysis we conducted. We tune the reaction function ψ according to Table 1.

QCR with and without mandate routing.

Figure 3 illustrates the need to implement mandate routing in query based replication. It was obtained for the power function with $\alpha = 0$. This result is representative of all comparison where mandate routing was turned on and off. As the time of the simulation evolves we see that the utility (as estimated in expectation on (a), and observed from real fulfillment in (b)) dramatically decreases with time when QCR does not implement mandate routing. Further investigations have shown that simultaneously the amount of mandate diverges for item less frequently requested. We see on (d), where the number of replicas is shown for the five most requested items, that QCR without mandate routing systematically overestimates their share and sometimes. In contrast, the number of replicas with mandate routing fluctuates around the targeted value, and QCR quickly converges and stay near optimal utility.

Comparison with fixed allocations.

Figure 4 presents the utility obtained with the both QCR and the competitor algorithms described previously. For each algorithm, we plot in the y-axis $(U - U_{\text{opt}})/|U_{\text{opt}}|$ where U is the utility obtained on average during the simulation by this algorithm and U_{opt} is the value obtained with the optimal allocation. Hence the plotted quantity is always negative (since as we expect no algorithm outperforms OPT). Value $y = -1$ corresponds to a utility 1% smaller than the optimal social welfare. Due to large variation of this quantity over the space and algorithms investigated, we used a logarithmic scale in the y-axis to present these results. For each algorithm, we consider two models of delay-utility (power and step function) with different parameters, varied along the x-axis.

We observe that for both delay-utility functions, the extreme strategies (*i.e.,* UNI and DOM) fail to approach the optimal in general. In particular it is the case for small value of α, when users are sensitive to waiting delay and the decrease in social welfare can be high, and small value of τ where quick response is essential. While demand aware offline strategies (*i.e.,* PROP and SQRT) perform similarly to QCR, QCR does not require control-channel information to achieve this performance. We even observe that QCR outperforms PROP in many cases, sometimes very significantly. Across all heuristic competitors, QCR does not incur a loss of utility beyond 5% (for step function) and 60% in the worst case of power function. One unexpected result is that the square root allocation performs reasonably well in most cases studied, however this is an ideal performance observed when the allocation is fixed with *a priori* knowledge. In contrast, proportional allocation leads to much worse performance, in particular for power delay-utility function. Proportional allocation resembles a passive demand based replication where a fixed number of replicas (*e.g.,* one replica) are created whenever a request is fulfilled (as found in [14] and many other works). These results illustrate that such passive replication simply gives too much weight to popular items, and that compensating for this effect is both necessary and achievable.

6.3 Real Contact Traces

We now abandon the homogenous mixing assumption needed for our analysis and look at the performance of QCR on real-world contact traces to see if the spirit that our analysis still applies under more realistic mobility. As in the homogenous experiments, we use $I = 50$ and $N = 50$ for evaluation of our techniques on both heterogenous traces.

Conference scenario.

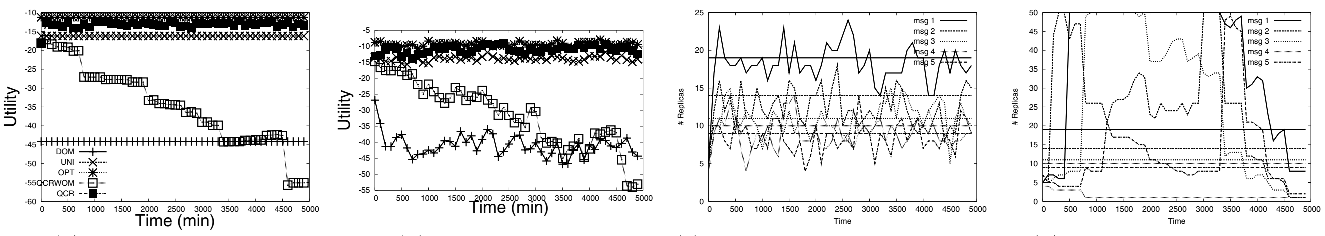
We use the Infocom '06 data set which measures Bluetooth sightings between 73 participants at the Infocom conference (see [4] for more details) over the course of thee days. To remove bias from poorly connected nodes, we selected the contacts for the 50 participants (numbered from 21 to 71 in the original data sets) with the longest measurement periods.

Figure 5 (a) presents the utility as seen over time (time averaged over an hour) for the competitor set and QCR (with mandate routing). We clearly observe the alternation of daytime and nighttime during the trace. Here, unlike in the homogenous scenario, DOM and PROP perform the best. QCR performs very close to the latter, despite heterogeneity and complex time statistics. SQRT and UNI perform poorly until tau becomes quite large - as the delay requirement is too stringent to allow significant improvement on non-popular items that would offset the loss created by shifting the focus off popular content.

Figure 5 (b) and (c) presents the relative loss of utility for different algorithms (compared with OPT) as a function of τ. We separate the impact of heterogeneity *per se* by presenting the actual traces and a synthetic trace where contact rates of all pairs are identical but contacts are assumed to follow memoryless time statistics. Heterogeneity *per se* does not seem to greatly impact the performance of QCR. Indeed it appears QCR may even perform better under contact complexity, perhaps because its implicit reaction to content availability adapts well to heterogeneous cases. The most notable difference with the homogeneous case is that SQRT is not a clear winner anymore and that PROP and DOM seem relatively stronger. The results from actual traces show that time statistics greatly impact the performance of a fixed allocation. First, since OPT was computed under the approximation of memory less contact, some competitors actually perform slightly outperform OPT on occasion. We also observe that the DOM greatly improves due to bursty statistics. However, the performance of QCR remains quite comparable, generally lying within 15% of OPT.

Vehicular networks.

We use contacts recorded between 50 taxicabs selected from the Cabspotting project contact traces. The data sets was extracted from a day of data and assumed that taxicabs are in contacts whenever they are less than 200m apart (see [3] for more details). Results, shown in terms of performance relative to OPT, may be found in Figure 6 (a) (b) (c). Again, we observe that OPT, which is based on a memoryless assumption, can be outperformed by some allocation (as in (b) for the step function case). Just as for the Infocom data set, we see that SQRT tends to produce degraded performance, while DOM improves as heterogeneity and complex time statistics are included in the contact trace. The performance of QCR, the only scheme based on local information, appears less affected by this change.



(a) Expected Utility (b) Observed Util. (c) Cache: Mandate Routing (d) no Mandate Routing

Figure 3: Effect of Mandate Routing (homogenous contacts, power delay-utility function with $\alpha = 0$).

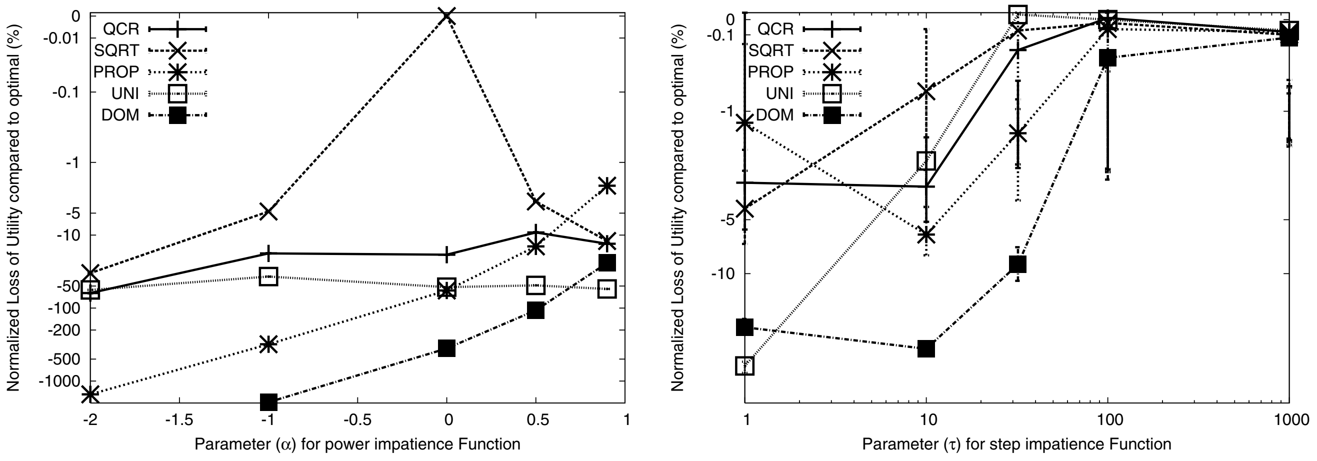


Figure 4: Comparison between QCR and several fixed allocations (homogeneous contacts): for power delay-utility function as a function of α (left), for step delay-utility function as a function of τ (right).

7. CONCLUSION

Our results focus on a specific feature which makes P2P caching in opportunistic network unique: users' impatience. From a theoretical standpoint, we have shown that optimality is affected by impatience but can be computed and moreover satisfies an equilibrium condition. From a practical standpoint, we have seen that it directly affects which replication algorithm should be used by a P2P cache. Passive replication, ending in proportional allocation, can sometimes perform very badly, but one can tune an adaptive replication scheme to approach the performance of the optimal, based only on local information.

We believe these results may serve as a stepping stone to address other unique specific characteristics of P2P caching in opportunistic system, in particular they offer a reference case from which one can study (1) the impact of heterogeneity and complex mobility property more systematically, (2) clustered and evolving demands in peers, as distributed mechanism like QCR naturally adapts to a dynamic demand. Another important aspect that remains to be addressed is how to estimate the delay-utility function implicitly from user feedback, instead of assuming that it is known.

8. ACKNOWLEDGMENT

We would like to gratefully acknowledge Nikodin Ristanovic, Stratis Ioannidis and Laurent Massoulié, for their insightful comments and their help during the preparation of this work.

9. REFERENCES

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN Routing as a Resource Allocation Problem. In *Proc. ACM SIGCOMM*, 2007.

[2] C. Boldrini, M. Conti, and A. Passarella. Contentplace: social-aware data dissemination in opportunistic networks. In *Proc. ACM MSWiM*, 2008.

[3] A. Chaintreau, J.-Y. L. Boudec, and N. Ristanovic. The age of gossip: spatial mean field regime. In *Proc. of ACM SIGMETRICS*, 2009.

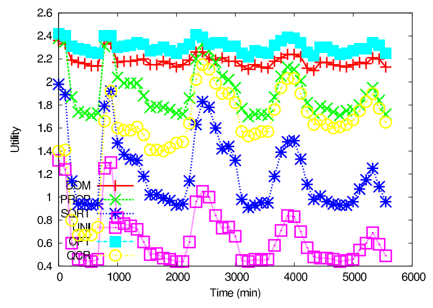
[4] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, J. Scott, and R. Gass. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Trans. Mob. Comp.*, 6(6):606–620, 2007.

[5] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.

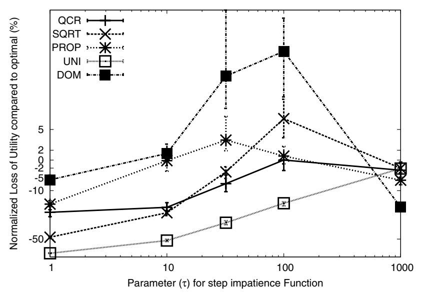
[6] P. Costa, C. Mascolo, M. Musolesi, and G. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE Jsac*, 26(5):748–760, June 2008.

[7] J. Greifenberg and D. Kutscher. Efficient publish/subscribe-based multicast for opportunistic networking with self-organized resource utilization. In *Proc. AINAW*, 2008.

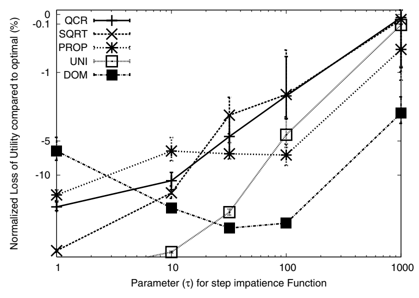
[8] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. on Net.*, 10(4):477–486, 2002.



(a) evolution with time ($\tau = 1$)

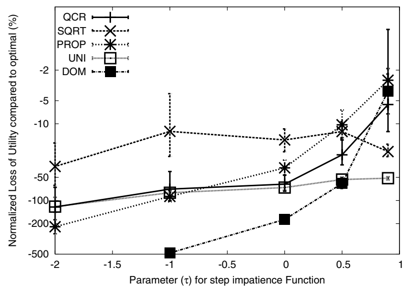


(b) vs. τ (actual trace)

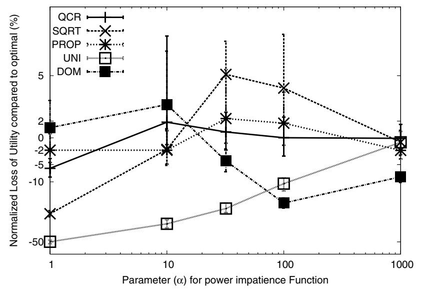


(c) vs. τ, (synthesized trace)

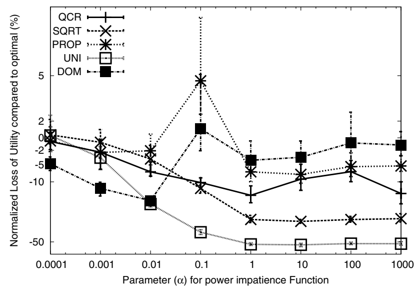
Figure 5: Utility for Infocom06 Dataset and Step Function Model of Impatience



(a) vs. α (Power fn)



(b) vs. τ (Step fn)



(c) vs. ν (Exponential fn)

Figure 6: Comparison between QCR and several fixed allocations (Cabspotting dataset using actual traces): for power delay-utility function as a function of α (left), for step delay-utility function as a function of τ (middle), for exponential delay-utility function as a function of ν (right).

[9] L. Hu, J.-Y. L. Boudec, and M. Vojnovic. Optimal channel choice for collaborative ad-hoc dissemination. Technical Report MSR-TR-2009-26, MSR, 2009.

[10] S. Jain, M. Demmer, R. Patra, and K. Fall. Using redundancy to cope with failures in a delay tolerant network. *ACM SIGCOMM Computer Communication Review*, 2005.

[11] G. Karlsson, V. Lenders, and M. May. Delay-tolerant broadcasting. *IEEE Transactions on Broadcasting*, 53:369–381, 2007.

[12] A. Krifa, C. Barakat, and T. Spyropoulos. Optimal buffer management policies for delay tolerant networks. *Proc. of IEEE SECON*, 2008.

[13] S. Kunniyur and R. Srikant. End-to-end congestion control schemes: utility functions, random losses and ecn marks. *IEEE/ACM Trans. Netw.*, 11(5), 2003.

[14] V. Lenders, M. May, and G. Karlsson. Wireless ad hoc podcasting. In *Proc. IEEE SECON*, 2007.

[15] C. Lindemann and O. P. Waldhorst. Modeling epidemic information dissemination on mobile devices with finite buffers. In *Proc. ACM Sigmetrics*, 2005.

[16] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mobile Computing and Communication Review*, 7(3), 2003.

[17] P. Liu, R. Berry, and M. Honig. Delay-sensitive packet scheduling in wireless networks. *Proc. of WCNC 2003*, 3, 2003.

[18] M. Musolesi and C. Mascolo. Car: Context-aware adaptive routing for delay-tolerant mobile networks. *IEEE Transactions on Mobile Computing*, 2009.

[19] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 1978.

[20] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proc. ACM MobiHoc*, 2001.

[21] J. Reich and A. Chaintreau. The age of impatience: optimal replication schemes for opportunistic networks. Technical Report CR-PRL-2009-06-0001, Thomson, 2009. available at: www.thlab.net/~chaintre/pub/reich09age.TR.pdf.

[22] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. In *Proc. ACM MobiCom*, 2006.

[23] G. Sollazzo, M. Musolesi, and C. Mascolo. Taco-dtn: a time-aware content-based dissemination system for DTN. In *Proc. ACM MobiOpp*, 2007.

[24] T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient routing in intermittently connected mobile networks: the single-copy case. *IEEE/ACM Trans. on Netw.*, 16(1), Feb 2008.

[25] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *Proc. INFOCOM*, 2006.

[26] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for pub/sub communication in DTN. In *Proc. ACM MSWiM*, 2007.