Contextual Bandits in P2P Systems

N. Bui

February 29, 2016

Abstract

A P2P file distribution system like BitTorrent is comprised of peers downloading the same file at the same time while uploading pieces of the file to each other. Policies for resource allocation make each peer's download rate proportional to their upload rate. Since each peer is responsible for maximizing its own download rate, peers would seemingly upload to the peers that provide the best download rate. But this doesn't address the issue that there are unused connections to undiscovered peers that could provide a better download rate. By modeling each peer as a contextual bandit, peers have a more robust method for discovering peers with potentially better download rates. We will show that the bandit model naturally causes peers to aggregate in groups that maximizes download rates for group members. We use an agent-based modeling approach in the simulation to explore the system's behavior.

1 Introduction

Peer-to-peer (P2P) applications do not have a central authority. Peers form networks with other peers and share resources together creating inexpensive, highly scalable and robust platforms [1]. P2P file distribution systems like BitTorrent [2] redistributes the cost of uploading content to downloaders by offloading that cost to multiple peers. To create a system where peers are motivated to exchange resources, most P2P file distribution systems follow a tit-for-tat strategy. This strategy enforces behavior where a peer can only receive resources equivalent to the resources it has given. Studies based on game theoretic approaches assert that the resource reciprocation strategy can lead to a Nash Equilibria or pareto optimality for download rates amongst all peers [2] [1]. It has also been proposed that by modeling P2P as a Markov Decision Process (MDP), the long-term utilities, the download rates, are maximized [4] [5]. So within the constraints of a tit-for-tat strategy, peers estimate other how much resources they may potentially receive from another peer to determine how much resources they should allocate. The MDP addresses the resource reciprocation problem within a clearly defined group of peer but does not address the issue of finding new peers that may result in better download rates. The BitTorrent protocol has an exploration rule to force peers to allocate resources to a new peer regardless of the download rate from that peer in a round-robin fashion. The peer is periodically rotated and Cohen [2] describes the behavior as analogous to always cooperating on the first move in the prisoner's dilemma.

To address these limitations, we model each peer as containing a contextual bandit to explore and find the best peers to exchange resources with. The peers will still follow a "tit-for-tat" strategy only now with the additional ability to explore and find a potentially better peer. A simulation using agent-based modeling demonstrates that peers settle on groups that maximize each other's download rates. We show that peers can maximize their download rates in a realistic setting where the time horizon is not long. The bandit strategy avoids the issue of estimating state transitions and the storage of the numerous state descriptions as required in the MDP solution [4]. We will show that peers will naturally form groups that maximize each other's download rates due to autonomy of each peer striving to maximize their download rates. This avoids the determining the profit a peer can gain in joining an existing group [1].

2 Our Agent-based Model

In this section, we describe the P2P system model.

Peers in P2P systems congregate with other peers that possess content they desire. Consider an ecosystem of N peers where each peer contains content that all other peers need. So all N peers can potentially exchange resources with other peers. Each peer is identified as P_1, P_2, $\dots$, P_N. A peer chooses to allocate resources to other peers knowing that they must abide by the tit-for-tat policy and reciprocate resources. These peers will reciprocate a portion of their resources, the maximum upload bandwidth, that is scaled according to the received resources from that peer. For example, a peer P_i, where $i \in [1, N]$, receives a download rate of 30 Mbps from its peers. 30 Mbps is the sum of all the allocated upload bandwidth from its peers. Suppose a peer P_j, where $j \in [1, N]$ and $j \neq i$ is responsible for 10 Mbps of the total 30 Mbps. If peer P_i has a maximum possible upload bandwidth of 12 Mbps, peer P_i must reciprocate 4 Mbps to peer P_j. The reciprocation is proportional to the resources it received from peer P_j and the total amount it has received.

$12Mbps \times \frac{10Mbps}{30Mbps} = 4Mbps$

Peers do not have to reciprocate resources if the resources reciprocated are less than a systemwide defined minimum. This prevents peers from having their resources thinned-out from reciprocating too many peers. If a peer has to reciprocate to too many peers, its resources can effectively be dwindled to a point that adversely affects the quality of service. By creating a minimum requirement, we can guarantee a quality of service.

In a game-theoretic interpretation of this model, peers are self-interested because they will explore to maximize their download rate. Peers in this model and in the simulation will be heterogenous. Their maximum upload rates do not have to be equal.

The problem of finding a new peer or remaining loyal to the current reciprocation is naturally a classical exploration vs. exploitation scenario. This motivates the bandit solution proposed in the next section. As dictated by the bandit algorithm employed by each peer, a peer can choose to keep their reciprocation amongst the current peers or they can identify a new peer to give resources to, hoping that the subsequent reciprocation will be beneficial.

This paper will not discuss how pieces of content are selected and in what order they are selected. It is an important aspect affecting the performance of the system [2]. In this paper, we will assume it is adequately handled and is independent to the resource reciprocation's impact on the performance of the network.

3 Contextual Bandits

Each peer implements a contextual bandit algorithm to dictate the exploration and exploitation for resource allocation with new peers. The contextual bandit differs from the traditional multi-armed bandit setting because it takes advantage of the information inherent in the environment. In this paper, we use the LinUCB algorithm described by Li [3].

In the classical K-armed bandit setting, an agent interacts with the environment amongst a set of K arms. The world reacts by giving the agent a reward. The multi-armed bandit learns a policy to dictate whether it should exploit the best arm, based on the history of rewards, or explore other arms. Exploration is motivated by the hope of finding a rewarding arm. Most algorithms taper their exploration as time progress because evidence for the optimal arm is strongly supported by the long history of rewards. The algorithm should maximize its rewards by exploiting the best arm. This setting does not account for extra information that is typically available in the environment.

In the contextual bandit setting, an agent interacts with the environment given a context from the setting. This additional information should allow the agent to optimize arm selection based on the context. Consider a context vector $\mathbf{x}$ of length L where each component $x_i \in [0, L)$ corresponds to a feature of the context. To be robust, contextual bandit algorithms aim to handle an infinite vector-space of contexts. The arms are usually constrained and comprised of a predefined set.

4 Contextual Bandit as Peers

To model each peer as a contextual bandit, we assume that each peer knows all other peers with the content they desire but they do not know their maximum possible upload bandwidth. Peers will always reciprocate resources. That is, for every round, if another peer provides them resources, they must reciprocate. Only if by reciprocating resources causes them to give any peers less than the systemwide minimum allocation will a peer not reciprocate. This behavior creates a guaranteed quality of service since no peer should ever experience a download rate less than the systemwide minimum. However, since the employed policy is "tit-for-tat", a peer who has a maximum upload rate less than the systemwide minimum will be unable to join the system. This behavior is also responsible for peers to settling in subgroups where members subgroup members mutually benefit from each other and choose not to interact with other peers. The contexts in this scenario are the received resources from other peers. This context will be used to compute the resource reciprocations. Each arm is a peer that P_i can add to it's resource reciprocation. For each arm, a peer P_i will consider the estimated resources peer P_j may allocate in addition to the received resources from other peers. The hypothetical resource reciprocation is computed with this additional peer. Each peer tracks an estimate of other peers' resource reciprocation. This is tracked as an average. So if P_i has received 10 Mbps, 0 Mbps, and 2 Mbps from P_j in 3 different trials, P_i's estimated resources reciprocation from P_j will be 3 Mbps. These will be discussed as the features in the LinUCB algorithm.

4.1 Algorithm

The contextual bandit algorithm proceeds in discrete rounds, $t = 1, 2, 3, \ldots$. For round t:

1. The algorithm receives a state $\mathbf{S}_t$ where each component $s_{j,t} \in \mathbf{S}_t$ represents the resources received from peer P_j and a set $\mathcal{A}_t$ of arms with a feature vector $\mathbf{x}_{t,a}$ for each arm $a \in \mathcal{A}_t$. Both the state $\mathbf{S}_t$ and the feature vector $\mathbf{x}_{t,a}$ is referred to as the *context*.

2. Based on observed rewards in previous rounds, the peer chooses an arm $a_t \in \mathcal{A}_t$. It subsequently receives a reward $r_{a,t}$.

3. The algorithm updates its reward estimate from the observation $(\mathbf{x}_{t,a}, a_t, r_{a,t})$.

Algorithm 1 LinUCB as applied to P2P peers.

```
0: Inputs: $\alpha \in \mathbf{R}$
1: for $t = 1, 2, 3, \ldots, T$ do
2:     Observe resource reciprocation $\mathbf{S}_{i,t}$
3:     for all $P_j \in group, i \neq j$ do
4:         if no reciprocation estimate for $P_j$ then
5:             $\mathbf{A}_{P_j} \leftarrow \mathbf{I}_d$ ($d$-dim identity matrix)
6:             $\mathbf{b}_{P_j} \leftarrow \mathbf{0}_{d\times 1}$ ($d$-dim zero vector)
7:         end if
8:         $\hat{\theta}_{P_j} \leftarrow \mathbf{A}_{P_j}^{-1}\mathbf{b}_{P_j}$
9:         $\mathbf{x}_{t,P_j} \leftarrow$ reciprocation($\mathbf{S}_{i,t}, P_j.estAlloc$)
10:        $p_{t,P_j} \leftarrow \hat{\theta}_{P_j}^T \mathbf{x}_{t,P_j} + \alpha \sqrt{\mathbf{x}_{t,P_j^T} \mathbf{A}_{P_j}^{-1} \mathbf{x}_{t,P_j}}$
11:    end for
12:    Choose peer $P_j = \arg\max_{P_j \in group} p_{t,P_j}$ to add to reciprocation.
13:    Observe payoff $r_t$
14:    $\mathbf{A}_{P_j} \leftarrow \mathbf{A}_{P_j} + \mathbf{x}_{t,P_j}\mathbf{x}_{t,P_j}^T$
15:    $\mathbf{b}_{P_j} \leftarrow +r_t \mathbf{x}_{t,P_j}$
16: end for
```

4.2 Features for Contexts

A peer P_i will consider 2 features for every peer: the estimated resource reciprocation and resource to allocate. The algorithm will maximize rewards assuming rewards are correlated with monotonically increasing features. Certainly, a higher estimated resource reciprocation from a peer correlates with a higher reward. The other feature is not obvious at first.

We have an optimistic approach to resource reciprocation: the more we can give, the more likely we can receive in turn. That is the reasoning behind the second feature. A simple first choice for a feature could have been the "benefit" ratio: how we receive over how much we have to give. This would certainly favor peers that we can gain a lot of resources without giving up much of our own. It follows the monotonously-increasing requirement for the correlation of rewards with feature. But this leads to oddities in the simulations. Very low upload-rate peers continuosly spam high-bandwidth peers because the ratios explode when the peer's upload-rate is very small. Using the optimistic approach fits better with the "tit-for-tat" strategy.

It is worth mentioning the fallbacks of using an average for the estimated resource reciprocation for a peer. A peer can potentially gain more resources and thus, be of great value to other peers. However, it previously appeared as weak to other peers. To adjust the average in the eyes of the other peers, the newly-improved peer must work against a possibly large sample of resource reciprocations in order to change it's estimated resource reciprocation in the eyes of other peers.

Table 1: Peers maximum upload rate, in Mbps

Peer 0	28
Peer 1	18
Peer 2	15
Peer 3	23

5 Simulation Results

Simulations are configured in Python. At each round, each peer is allowed the opportunity to explore or exploit. Initially, all peers initialize their expected reciprocated resources from all other peers as the system minimum.

In this first simulation, we have a group of 4 peers, similar to the simulation found in [4] [5]. Their maximum upload rate is given by Table 1.

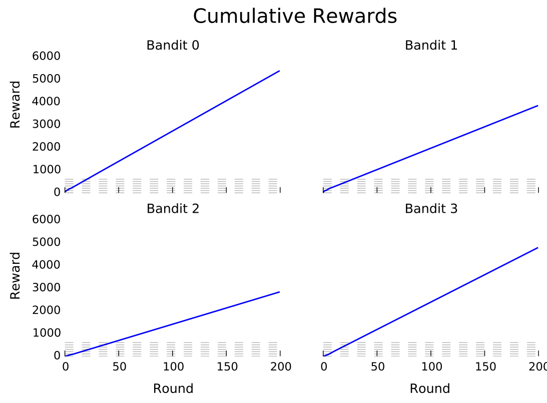

Figure 1: Cumulative Rewards for each peer

Figure 1 displays the cumulative reward of each peer in the group and Figure 2 displays the instantaneous reward per round for each peer.

Each peer goes through a period of exploring and then settling on an optimal resource reciprocation that maximizes their download rate. This period of exploration can be seen in the figures. In Figure 1, the periods of exploration result in varying rewards which manifests as lines of changing slope in the cumulative reward plot. In Figure 2, the periods of exploration result in varying rewards which manifests as noisy variances in the instantaneous reward plot. Many of the LinUCB algorithm traits can be identified in Figure 2. At steady-state, the resource reciprocation is described as an adjacency matrix in Table **??**. Because resources allocation is inherently a one-directional action, these edges are directed edges so it's important to interpret the rows and columns. When read by rows, each column represents a peer and how much resources it gives to the peer identified in the row. For example, Peer 0 receives an 18 Mbps download rate from Peer 1 and 8.7 Mbps from Peer 3. Note that the sum of each column equals the maximum that the peer can give.

Table 2: Steady-state Resource Allocation, in Mbps

	Peer 0	Peer 1	Peer 2	Peer 3
Peer 0	0	18	0	8.7
Peer 1	18.9	0	0	0
Peer 2	0	0	0	14.3
Peer 3	9.1	0	15	0

Most peers begin with 0 reward because no resources are allocated to it and thus they are not required to reciprocate resources with any other peers. Notice that P_0 begins with a non-zero amount of reward. At these early rounds, peers will favor exploring because they want to find reciprocating peers. Because of the deterministic nature of LinUCB's exploration, all peers explore P_0 first due to the index of P_0 arm in the set of arms. This is why immediately, bandit 0 has a nonzero instantaneous reward for round 0. Bandit 0, since it can't give resources to it-self, will give resources to the closest index, P_1, which is why P_1 also begins with a nonzero instantaneous reward. Bandit 1 also receives a spike, but not immediately, in reward due to collisions in exploring peers. Not all peers are satisfied by P_0 reciprocation and thus they explore the next arm which happens to be P_1. As rounds progress, each peer is updating its estimation of other peers' upload rates. Peers who can't reciprocate because they need to adhere to the systemwide required upload rate or give relatively low upload rates have lower estimated upload rates. A peer is likely to give resources to these peers knowing that they will receive little or no resources in return. They choose to exploit their best resource allocation. So then peers settle amongst a subgroup of peers whose resource reciprocation maximizes their download rate. They explore less and will only give resources to members of the subgroup.

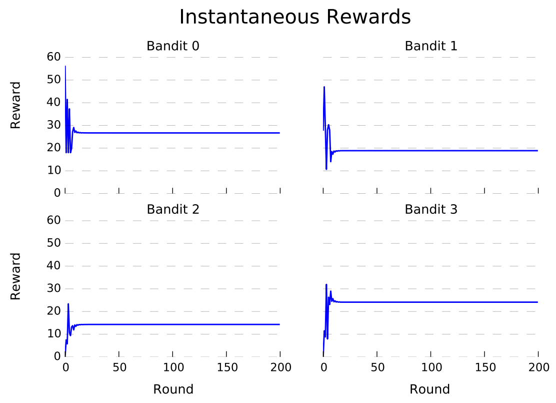

Figure 2: Instantaneous Rewards for each peer

5.1 Group Formation

Because as rounds progresses, peers often stay in the same subgroup because they have identified, with stronger confidence as more rounds progress, the most beneficial peers to maximize their download rates. This is more evident in a large group of peers. In Figure 3, a group of 50 peer settle into subgroups after 1000 rounds. Colored in blue are peers who do not receive any resources. If there are is an edge present with a blue peer, it means they are giving resources yet they are being reciprocated. Light blue peers have only one peer that is allocating them resources. Orange peers have more than one peer allocating them resources. Note also that there are peers that are alone. These are peers whose maximum possible upload rate is less than the system's required minimum. Since

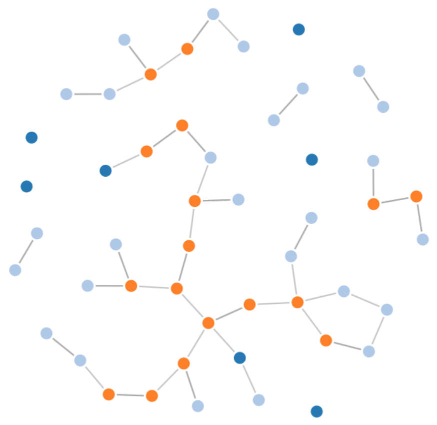

Figure 3: Subgrouping in a 50-Peer Group. *Blue*: not receiving resources. *Light Blue*: only one reciprocating peer. *Orange*: more than one reciprocating peer.

they can't satisfy the minimum required resource allocation, they receive none in return.

Using the contextual bandit algorithm for exploration, the problem of determining the benefit for a peer joining an existing group is avoided. Peers simply explore to find the optimal peers and groups will naturally form when they settle amongst a set of beneficial peers. Computing the benefit of joining an existing group as described by Buragohain et. al. [1] involves solving an $N \times N$ interaction matrix between heterogeneous peers by which no closed form solution exists.

6 Application

The contextual bandit creates a system where a minimum quality of service is guaranteed but not everybody can participate.

6.1 Robustness

Policies such as the differential service policy "tit-for-tat" are rule-based. The bandit algorithm alone is not enough to handle all the use cases and failure modes that occur in a P2P system. While the bandit algorithms improve upon the search of better peers, additional rules should be defined to improve the robustness of the P2P system. In this section, we will discuss the robustness of the P2P with only the contextual bandit algorithm and no other rules defined.

6.1.1 Joining Peer

Consider an existing group of peers in steady-state resource reciprocation and a new peer joins. Table 1 contains each peers' maximum upload rate. This is the maximum amount of resources each peer can give.

In this scenario, a fifth peer will join and its maximum upload rate is 10 Mbps. This peer is identified as "Peer 4". Their steady-state resource allocation is decribed in 2. Figure 4 shows the instantaneous rewards of the peer that joined the group. Note that the peer joined at round $t = 50$ so its reward, the download rate, is 0 up to that time. Similarly in Figure 5, notice the changes in the subgroup at round $t = 50$.

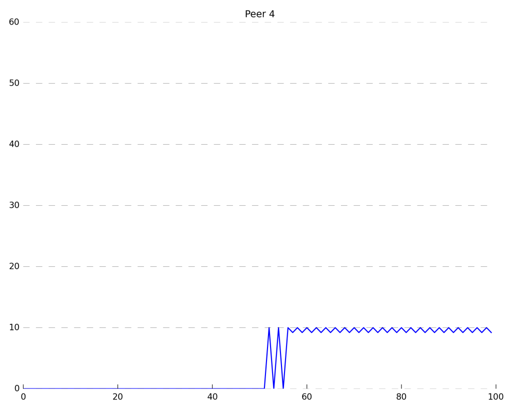

Figure 4: Instantaneous rewards of the bandit that joined the subgroup.

Table 3: New steady-state Resource Allocation 1 of 2, Mbps

	Peer 0	Peer 1	Peer 2	Peer 3	Peer 4
Peer 0	0	18	0	8.7	0
Peer 1	18	0	0	0	0
Peer 2	0	0	0	14.3	0
Peer 3	0	0	15	0	10
Peer 4	10	0	0	0	0

Up to $t = 50$, the group has settled on a steady-state resource allocation. When peer 4 joins, it perturbs the system. Notice in both Figure 4 and Figure 5, the resource reciproca-

Table 4: New steady-state Resource Allocation 2 of 2, Mbps

	Peer 0	Peer 1	Peer 2	Peer 3	Peer 4
Peer 0	0	18	0	0	10
Peer 1	18.9	0	0	0	0
Peer 2	0	0	0	13.8	0
Peer 3	9.1	0	15	0	0
Peer 4	0	0	0	9.2	0

tion does not settle and there is noise. The state flips between 2 states and they are shown in 3 and 4.

The reason for this fluttering of states is because the expected rewards of 2 different arms are close in value so the algorithm constantly switches between the two. For peer 4, it debates between peer 0's 10 Mbps or peer 3's 9.2 Mbps. For peer 0, it debates between peer peer 4's 10 Mbps or peer 3's 8.7 Mbps. For peer 3, it debates between peer 0's 9.1 Mbps or peer 4's 10 Mbps. Notice that other peers also receive resources from peer 4's exploration, though only momentarily. Peers 1 & 2 remain mostly the same since peer for can only offer them 10 Mbps, less than their current download rate of 18 and 14.3 Mbps respectively.

In a deployed system, this issue is undesirable because the changing of states can interrupt downloads. A new feature can be introduce which is the potential gain for leaving an incomplete download.

6.1.2 Dropping Peer

Consider an existing group of peers in steady-state resource reciprocation and a peer drops. Refer back to Table 1 which contains each peers' maximum upload rate. This is the maximum amount of resources each peer can give.

We consider the group before with all peers, peers 0 through 4. Their steady-state resource reciprocation can described by Table 4. Note that unlike in the previous scenario where the peer joins, the states do not flutter here even though they are similarly configured.

In this scenario, the fifth peer, Peer 4, will drop and its maximum upload rate is 10 Mbps. Table 5 shows the steady-state of the new system with one less peer. Notice that after, the peers form couples, only exchanging their resources with one other peer and giving that peer all their resources.

Table 5: New steady-state Resource Allocation, Mbps

	Peer 0	Peer 1	Peer 2	Peer 3
Peer 0	0	18	0	0
Peer 1	28	0	0	0
Peer 2	0	0	0	23
Peer 3	0	0	15	0

Figure 6 shows the instantaneous rewards of the peer that dropped from the group. Note that the peer dropped at round $t = 50$ so its reward, the download rate, is 0 after that time. Similarly in Figure 7, notice the changes in the subgroup at round $t = 50$. Peer 3 has the most significant loss because it was exchanging resources exclusively with peer 4. Notice that peer 3 occasionally explores peer 1 by giving it resources, more so as rounds progress. This is evident by peer 2's loss of rewards and peer 1's positive gain of rewards. However, peer 3 sees that peer 2 is the only once reciprocating because peer 1 exchanges only with peer 0.

Notice that the state described by 5 is different from the state described by 2.

With the addition of the bandit exploration,

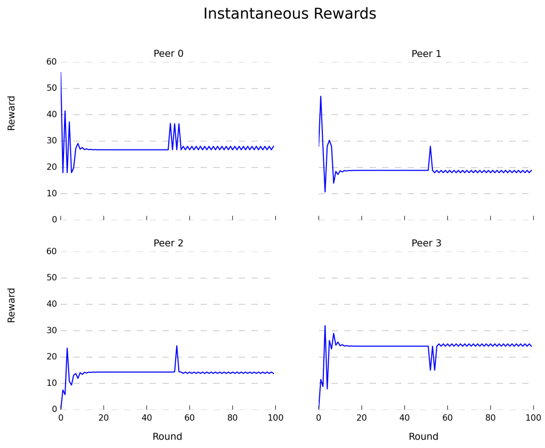

Figure 5: Instananeous rewards of the subgroup that experience a new peer joining.

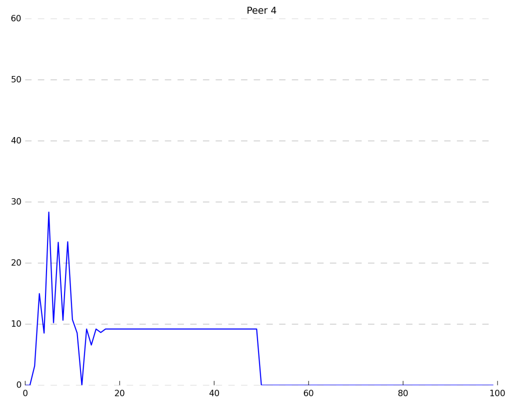

Figure 6: Instantaneous rewards of the bandit that dropped from the subgroup.

there should be additional rules that help the group (or subgroups) to recover from a dropped peer or take advantage of a new peer.

We assert that this robustness can be extended to cases where peers have changing maximum upload bandwidth that they can allocate.

7 Conclusion

While the MDP approach is successful, key issues are introduced when considering real user behaviors. In deployment, there are high churn rates because peers rarely connect for more than a few hours [2]. The shorter than expected time horizon also works in opposition to the estimation of other peers' behaviors. As proposed by Park [4], a peer can identify its state transition probabilities based on the history of resources reciprocation. A shorter time horizon impedes the quality of the estimation of state transition probabilities. The bandit algorithm does away with these issues and abstracts all those issues as a explore vs. exploit scenario.

References

[1] Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. A game theoretic framework for incentives in P2P systems. *CoRR*, cs.GT/0310039, 2003.

[2] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[3] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, abs/1003.0146, 2010.

[4] Hyunggon Park and M. van der Schaar. A framework for foresighted resource reciprocation in p2p networks. *Multimedia, IEEE Transactions on*, 11(1):101–116, Jan 2009.

[5] Hyunggon Park and M. van der Schaar. Evolution of resource reciprocation strategies in p2p networks. *Signal Processing, IEEE Transactions on*, 58(3):1205–1218, March 2010.

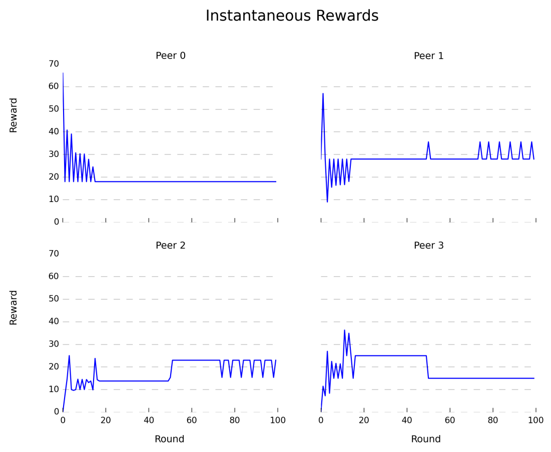

Figure 7: Instantaneous rewards of the subgroup that experience a peer dropout.