

Reverse-Engineering BitTorrent: A Markov Approximation Perspective

Ziyu Shao*, Hao Zhang⁺, Minghua Chen*, and Kannan Ramchandran⁺

*Department of Information Engineering

The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

Email: {zyshao, minghua}@ie.cuhk.edu.hk

⁺Department of Electrical Engineering and Computer Sciences, UC Berkeley, USA

Email: {zhanghao, kannanr}@eecs.berkeley.edu

Abstract—In this paper we understand BitTorrent protocol from a Markov approximation perspective. We show that together with the underlying rate control algorithm, the rarest first algorithm and choking algorithm in BitTorrent protocol implicitly solve a *cooperative* combinatorial network utility maximization problem in a distributed manner. This understanding allows us to access properties of BitTorrent from a fresh perspective, including performance optimality, convergence and impacts of design parameters. Our numerical evaluations validate the analytical results.

I. INTRODUCTION

In recent years the BitTorrent [1] has attracted the attention of both industry and academia, eager to understand the remarkable success of this simple yet powerful protocol. Why does BitTorrent work so well? How might, or should, it evolve in the future? Answers to these questions will help the fundamental understanding of BitTorrent, which not only help improve existing P2P systems in a systematic way rather than the way with ad-hoc heuristics, but also provide useful ingredients for synthesizing distributed algorithms for other combinatorial network design problems.

A lot of efforts [2] have been made to answer these questions, including real data measurements, game-theoretic analysis and differential-equation based macroscopic analysis. Through these efforts, it is clear now that piece and neighbor selection strategies are the two keys of efficient and scalable P2P systems. It is observed that rarest first algorithm, a piece selection scheme, guarantees close to ideal diversity of the pieces among peers; and choking algorithm, a neighbor selection scheme, provides an effective sharing incentive by implementing a tit-for-tat mechanism. Overall, BitTorrent is shown to be remarkably robust and scalable at ensuring high uplink bandwidth utilization [2], [3]. These observations and insights made by existing efforts enrich the fundamental understandings of BitTorrent and greatly sharpen the design skills of P2P systems.

Encouraged by the results from these exciting work, we further explore answers to the questions and reverse engineer BitTorrent from a new perspective, with the hope to reveal new hidden facts and provide new insights for future design. Our new perspective is based on the Markov approximation framework [4]. The main results are summarized in Table I. Detailed statements of results and contributions are listed as

follows:

- We find that with underlying rate-control algorithms such as TCP, BitTorrent actually implements a Markov chain solving the following global optimization problem: maximize the aggregate downloading rates of all peers given the underlying physical edge capacity constraints and concurrent uploading connections limit. BitTorrent solves this problem by combining three components (rate control, piece selection and neighbor selection) in three separated time scales respectively.
- More precisely, we consider all peer neighboring configurations satisfying concurrent uploading connections limit. First, the rate control algorithm assigns overlay edge capacity given the peer neighboring configuration and arbitrary underlying physical edge capacity constraints. By doing so, BitTorrent actually goes beyond a common assumption made in almost all P2P algorithm designs (except [5], [6]) that uplink and (or) downlink of peers is the only rate-limiting bottleneck. Second, rarest first algorithm, the piece selection component of BitTorrent, implicitly maximizes the aggregate downloading rates of all peers given the peer neighboring configuration and overlay edge capacity. Third, choking algorithm, the neighbor selection component of BitTorrent, implicitly finds the best peer neighboring configuration by implementing a Markov chain over all configurations and statistically hopping towards the best configuration.
- We characterize the following properties of the corresponding Markov chain: approximation gap, perturbation error bound, insensitivity of count-down time, mixing time, and trade-off between approximation gap and mixing time. These studies enable us to analyze properties of BitTorrent protocol including performance optimality, convergence and impacts of design parameters. Proofs and details are provided in [10].

II. MODELING BITTORRENT

A. Notations

Consider an underlying physical network modeled as a directed graph $G = (\mathcal{N}, \mathcal{L})$, where \mathcal{V} is the set of all physical nodes, including peering nodes and other intermediate nodes such as routers, and \mathcal{L} is the set of all physical links. Each link $l \in \mathcal{L}$ has a nonnegative capacity C_l .

TABLE I
SUMMARY OF RESULTS FOR REVERSE ENGINEERING BITTORRENT(BT) PROTOCOL

BT Components	Functionality	Time Scale	Critical Issues	Proof Techniques	Sections and References
Rate Control	Assigning overlay edge rates	Fast	Optimality	NUM	Sec. III-C-1) & [7]
			Convergence	Lyapunov Function	[7]
Piece Selection*	Maximizing the aggregate downloading rates of all peers	Normal	Optimality	Linear Programming	Sec. III-C-2) & [7]
			Convergence	Lyapunov Function	[7]
Neighbor Selection	Choosing the best peer neighboring configuration	Slow	Optimality	Markov approximation and perturbation analysis	Sec. III-C-3), IV & [4], [8]
			Convergence	Markov approximation and mixing time	Sec. III-C-3), IV-G & [4], [9]

* Note that our formulation is complementary to existing formulations of piece selection component [2].

Consider a P2P file sharing system over G . Note that in this paper, we do *not* consider dynamic scenarios, i.e., peers come and go. We use $V \subseteq \mathcal{N}$ to denote the set of all peering nodes. We use E to denote the set of directed overlay links between these peers. Note an overlay link (u, v) means u can send data to v by setting up connections. For all $e \in E$ and $l \in \mathcal{L}$, we define

$$a_{l,e} = \begin{cases} 1, & \text{if overlay link } e \text{ passes physical link } l; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

For any edge $e \in E$, we denote z_e as the associated edge rate. We denote E_f is the edge set of overlay graph under f . We also denote $\mathbf{z}_f = [z_e, e \in E_f]^T$ and $\mathbf{a}_l = [a_{l,e}, e \in E_f]^T$.

Each peer v has a neighbor set denoted by N_v and the size of N_v is denoted by Δ . However, each peer v can upload to at most δ neighbors simultaneously. We denote f as peer neighboring configuration, i.e., a specific peer neighboring relationship satisfying limits on the number of concurrent uploading connections. Let N_v^f denote the active neighbor set of peer v under configuration f . We also denote \mathcal{F} as the set of all possible peer neighboring configurations. For any peer $u \in V$, let x_{uv}^f denote the peer u 's downloading rate from peer v under f , and $\mathbf{x}^f = \{x_{uv}^f\}$ denote the vector of peers' downloading rates given configuration f .

B. Our Perspective

We adopt a deterministic fluid model. Overall, together with TCP rate control algorithms, BitTorrent solves a global optimization problem: given the underlying physical capacity constraints and concurrent uploading connections limit, maximize the aggregate downloading rates of all peers. This global optimization problem is formulated as follows:

$$\max_{f, \mathbf{x}^f > \mathbf{0}} \sum_{u \in V} \sum_{v \in N_u} x_{uv}^f \quad (2)$$

$$\text{s.t.} \quad \mathbf{x}^f \in \Gamma(\tilde{\mathbf{z}}_f), \quad (3)$$

$$f \in \mathcal{F}. \quad (4)$$

where $\Gamma(\tilde{\mathbf{z}}_f)$ is the feasible region for downloading rates \mathbf{x}^f given configuration f and overlay edge rates $\tilde{\mathbf{z}}_f$, and $\tilde{\mathbf{z}}_f$ is the optimal solution of the following rate-control problem:

$$\text{RC} : \max_{\mathbf{z}_f \geq \mathbf{0}} U(\mathbf{z}_f) \quad (5)$$

$$\text{s.t.} \quad \mathbf{a}_l^T \cdot \mathbf{z}_f \leq C_l, \forall l \in \mathcal{L} \quad (6)$$

Here $U(\cdot)$ is the TCP utility function, and the constraint in (6) is the physical link capacity constraint.

The above global optimization problem is challenging to solve due to its combinatorial nature. BitTorrent solves this global optimization problem by combining three components (rate control, piece selection and neighbor selection) with three

time scales (T_{rs}, T_{ps}, T_{ns}) respectively. The smaller the time scale, the faster the corresponding component converges. We make the time scale separation assumption, represented by $T_{rs} \ll T_{ps} \ll T_{ns}$.

1) *Rate Control Component*. The purpose of this component is to determine the overlay link rate given the overlay configuration f . This part is formulated as problem **RC**. By [7] we know that TCP protocol is reverse engineered to solve the problem **RC**. More specifically, TCP protocol allocates overlay edge rates subject to underlay edge capacity according to TCP utility maximization. For example, when TCP Reno protocol is adopted, $U(\mathbf{z}_f)$ is shown to be $\sum_{e \in E_f} \frac{1}{z_e d_e}$ [7], where d_e is some delay metric on link e . The proofs for both optimality and convergence to optimality are standard routine by adopting Lyapunov technique [7].

Remarks: In practice, some ISPs (internet service providers) such as Comcast started to throttle the BitTorrent Traffic, creating bottlenecks at ISP peering points. Moreover, the capacity bottleneck can be anywhere in the network, not necessarily at the edge of the network [11]. Our above model and formulation allows us to access the performance of BitTorrent over arbitrary network topologies where bottlenecks can be anywhere in the network. This is different from most other P2P models that assume the peer uplinks/downlinks are the only rate-limiting bottleneck.

2) *Piece Selection Component*. The purpose of this component is to maximize the summation of aggregate downlink rates of all peers given the overlay configuration f and overlay edge rates $\tilde{\mathbf{z}}_f$. This part is formulated as follows:

$$\text{PS} : \max_{\mathbf{x}^f > \mathbf{0}} \sum_{u \in V} \sum_{v \in N_u} x_{uv}^f \quad (7)$$

$$\text{s.t.} \quad \mathbf{x}^f \in \Gamma(\tilde{\mathbf{z}}_f) \quad (8)$$

This is the linear programming problem and we can adopt standard Lyapunov technique [7] to show optimality and convergence of corresponding subgradient algorithms.

How to relate this fluid flow model with the scheduling of discrete file pieces? In our opinion, for every downloading rate computed by PS component, piece selection algorithms in BitTorrent such as rarest first strategy and others [2], [3] aim at achieving such downloading rate by carefully exchange file pieces so that every receiving bit is ensured to be useful for the receivers. It remains open to investigate the correctness of this conjecture. In this paper, we focus on fluid model based study to reveal the big picture and in particular understand the neighbor selection in BitTorrent. The investigation on the above conjecture is left for future work.

Note that there are several other formulations for piece selection problems [2]. Our formulation is complementary to

existing models.

3) *Neighbor Selection Component*. Suppose the optimal solution to problem **PS** given configuration f is denoted by $g_f = \sum_{u \in V} \sum_{v \in N_u} x_{uv}^{*f}$, which represents the system utility under f . The purpose of this component is to find the best configuration such that its corresponding system utility g_f is maximum. This part is formulated as follows:

$$\text{NS} : \max_{f \in \mathcal{F}} g_f \quad (9)$$

This is a combinatorial optimization problem and we have the following result:

Proposition 1. *In general, for any $\delta \geq 2$, the problem NS is NP-complete and APX-hard (no effective polynomial-time approximate solution).*

III. NEIGHBOR SELECTION OF BITTORRENT

By Markov approximation framework [4], the problem of **NS** are solved approximately by designing a Markov chain in a distributed way.

A. Markov Approximation Framework

Original paper in [4] explains the framework from the optimization perspective. Here we present the framework from the sampling perspective.

Without loss of generality, we assume that the optimal solution for problem **NS** is unique and denote it as follows:

$$f^o = \arg \max_{f \in \mathcal{F}} g_f \quad (10)$$

We associate with each configuration $f \in \mathcal{F}$ a probability p_f . Then we can see that solving problem **NS** is equivalent to sampling the configuration space \mathcal{F} from the following Dirac distribution:

$$p_f = \begin{cases} 1 & \text{if } f = f^o \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

However, Dirac distribution is hard to obtain since f^o is unknown to us. Therefore, we need to sample the configuration space \mathcal{F} from a new target distribution, which needs to satisfy two conditions:

- **C1**: it can be obtained without knowing the exact value of f^o .
- **C2**: f^o is the configuration with the largest probability (not necessary to be 1).

It turns out a product-form distribution parameterized by $\beta > 0$ is a choice of this target distribution, shown as follows:

$$p_f^*(g) = \frac{\exp(\beta g_f)}{\sum_{f' \in \mathcal{F}} \exp(\beta g_{f'})}, \quad \forall f \in \mathcal{F}, \quad (12)$$

As we will see later that this product-form distribution can be obtained by designing a time-reversible Markov chain without knowing the value of f^o . On the other hand, given a positive constant β , it is not hard to see $f^o = \arg \max_{f \in \mathcal{F}} p_f^*(g)$. Thus both conditions **C1** and **C2** are satisfied.

Then when we sample the configuration space \mathcal{F} from the distribution $p_f^*(g)$ in (12), i.e., time-sharing among different configurations according to distribution $p_f^*(g)$ in (12), we actually solve the problem **NS** approximately and obtain a close-to-optimal system utility.

In the following, we design a Markov chain to satisfy the above requirement and its construction requires global information of P2P systems. We call it “*perfect Markov chain*” and use it as a benchmark for further performance comparison.

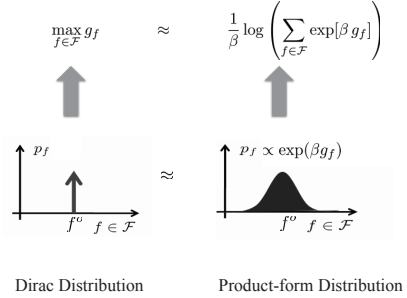


Fig. 1. Illustration of log-sum-exp approximation and sampling. We show the sampling distribution $p_f, \forall f \in \mathcal{F}$ and the corresponding performance metric $\sum_{f \in \mathcal{F}} p_f \cdot g_f - \frac{1}{\beta} \sum_{f \in \mathcal{F}} p_f \log p_f$, i.e., average of system utility $\sum_{f \in \mathcal{F}} p_f \cdot g_f$ off by an entropy item $\frac{1}{\beta} \sum_{f \in \mathcal{F}} p_f \log p_f$.

B. Perfect Markov Chain Design

First, we design the topology structure of state space. Direct transitions between two configurations $f, f' \in \mathcal{F}$ can happen if and only if:

- there $\exists \tilde{f}$ such that $\tilde{f} = f \cup f'$, $\tilde{f} \supseteq f$, $\tilde{f} \supseteq f'$, $|\tilde{f} \setminus f| = |\tilde{f} \setminus f'| = 1$;
- Link $\tilde{f} \setminus f$ and link $\tilde{f} \setminus f'$ originates from the *same* peer, denoted as $v(f, f')$.

We call the above conditions as *direct transition condition*. For any two configurations $f, f' \in \mathcal{F}$ satisfying direct transition condition, and any node $w \in \tilde{f} = f \cup f'$, we define

$$\mathcal{A}_{w, \tilde{f}} = \left\{ f'' \in \mathcal{F} \mid f'' = \tilde{f} \setminus \{(w, u)\}, \forall u \in \mathcal{N}_w^{\tilde{f}} \right\}. \quad (13)$$

We can see that any configuration f'' can be reached from \tilde{f} by node w choking one active neighbor in $\mathcal{N}_w^{\tilde{f}}$. Then for any two configurations $f, f' \in \mathcal{F}$ satisfying direct transition condition, we set the transition rates as:

$$q_{f, f'} = \tau \frac{\exp(\beta(g_{f'} - g_{\tilde{f}}))}{\sum_{f'' \in \mathcal{A}_{v(f, f'), \tilde{f}}} \exp(\beta(g_{f''} - g_{\tilde{f}}))} \quad (14)$$

where $\tau > 0$ is a constant and $\tilde{f} = f \cup f'$. It can be shown that the designed perfect Markov chain is a time-reversible Markov chain with the desired stationary distribution in $p_f^*(g)$ in (12).

However, one drawback of the design above is that it requires every peer $v \in V$ to know global information $g_{f''} - g_{\tilde{f}}$ for every adjacent configuration $f'' \in \mathcal{A}_{v, \tilde{f}}$.

C. BitTorrent Markov Chain Design

In this subsection, we modify direct transition rates in (14) by using local measurement quantity $-x_{\tilde{f} \setminus f'}$ to replace global information quantity $g_{f''} - g_{\tilde{f}}$. Thus for any two configurations $f, f' \in \mathcal{F}$ satisfying direct transition condition, the perturbed transition rates are

$$q_{f, f'} = \tau \frac{\exp(-\beta x_{\tilde{f} \setminus f'})}{\sum_{f'' \in \mathcal{A}_{v(f, f'), \tilde{f}}} \exp(-\beta x_{\tilde{f} \setminus f''})} \quad (15)$$

where $\tau > 0$ is a constant and $\tilde{f} = f \cup f'$.

This lead to a fully distributed implementation named as “soft choking algorithm” shown in Algorithm 1. Consequently, we no longer implement the perfect Markov chain but a perturbed one which we call “BitTorrent Markov chain”.

Algorithm 1: Soft Choking Algorithm

- The following procedure runs on each individual peer independently. We focus on a particular peer $v \in V$.
- **Initialization:** Peer v randomly selects δ neighbors from its neighbor list N_v and builds connections with these selected neighbors.
- **Step 1:** Denote by f the current configuration. Peer v independently generates an exponentially distributed random number with mean $\frac{1}{\tau(\Delta-\delta)}$ and counts down according to this number.
- **Step 2:** When the count-down expires, peer v random uniformly unchokes a new inactive neighbor w from N_v potential set. This happens with probability $\frac{1}{\Delta-\delta}$, and the system transits to a temporary configuration \tilde{f} .
- **Step 3:** Peer v chokes an active neighbor u with probability $\frac{\exp(-\beta x_{vu}^{\tilde{f}})}{\sum_{u' \in N_v^{\tilde{f}}} \exp(-\beta x_{vu'}^{\tilde{f}})}$, where $f' = \tilde{f} \setminus \{(v, u)\}$. Peer v then repeats **Step 1**.

D. Mapping Algorithm 1 to BitTorrent Protocol

We now map Algorithm 1 (the implementation of BitTorrent Markov chain) to BitTorrent protocol.

First, Step 1 of Algorithm 1 is exactly the same as the initialization of BitTorrent protocol.

Second, Step 2 of Algorithm 1 is nearly the same as the **optimistic unchoking** algorithm of BitTorrent protocol. The difference lies in the distribution of counter-time. The BitTorrent protocol used in practice adopts a constant count-down time, while Algorithm 1 adopts exponential distribution of count-down time. As it will see later in subsection III-F, the stationary distribution is insensitive to the distribution of count-down time. In this sense, we can say Step 2 of Algorithm 1 is performing optimistic unchoking scheme.

Third, Step 3 of Algorithm 1 (Soft-Worst-Neighbor-Choking) is a generalization of the **Choking** algorithm of BitTorrent protocol. In fact, as $\beta \rightarrow \infty$, we can see that with probability 1, peer v chokes the neighbor with the worst downloading rates. Thus, **soft-worst-neighbor-choking** scheme degenerates to **worst-neighbor-choke** scheme in BitTorrent as $\beta \rightarrow \infty$.

After building mapping from Algorithm 1 to BitTorrent Protocol, we can investigate properties of BitTorrent protocol by studying the properties of the corresponding Markov chain, including performance optimality, convergence and impacts of design parameters.

E. Impacts of Local Perturbation

For any direct transition from $f \in \mathcal{F}$ to $f' \in \mathcal{F}$, the perturbation error is defined as

$$\omega_{f,f'} = [g_{f'} - g_{\tilde{f}}] - (-x_{\tilde{f} \setminus f'}) \quad (16)$$

where $\tilde{f} = f \cup f'$.

Recall that $g_{(\cdot)}$ is the summation of all peers' aggregate downloading rates given configuration and f' is obtained from

\tilde{f} by dropping one link $\tilde{f} \setminus f'$. Therefore, $0 \leq g_{\tilde{f}} - g_{f'} \leq x_{\tilde{f} \setminus f'}$, and $\omega_{f,f'} \geq 0$. Without loss of generality, we assume $\omega_{f,f'}$ is bounded and takes values between 0 and Λ_{\max} .

Perturbation errors are incurred by replacing global knowledge $[g_{f'} - g_{\tilde{f}}]$ with local estimation $-x_{\tilde{f} \setminus f'}$, thus the stationary distribution of BitTorrent Markov chain is different from the one of Perfect Markov chain. Based on a recently developed analysis on two-dimensional perturbation errors [12], we have the following result:

Theorem 1. (a) *The stationary distribution of BitTorrent Markov chain is*

$$\bar{p}_f(\mathbf{g}) = \frac{\sigma_f \exp(\beta g_f)}{\sum_{f' \in \mathcal{F}} \sigma_{f'} \exp(\beta g_{f'})}, \forall f \in \mathcal{F} \quad (17)$$

where $\sigma_f = \sum_{k=0}^{n_f} \rho_{f_k} \exp\left(\beta \frac{k \Lambda_{\max}}{n_f}\right)$, $n_f, \forall f \in \mathcal{F}$ is the level of quantization errors and ρ_{f_k} ($0 \leq k \leq n_f$) is the distribution of quantized perturbation errors.

(b) Let $g_{\max} = \max_{f \in \mathcal{F}} g_f$ denote the optimal system utility, $g_{ave}^* = \sum_{f \in \mathcal{F}} p_f^* \cdot g_f$ denote the expected system utility with perfect Markov chain, and $\bar{g}_{ave} = \sum_{f \in \mathcal{F}} \bar{p}_f \cdot g_f$ denote expected system utility with BitTorrent Markov chain. Then the optimality gap are shown as follows:

$$0 \leq g_{\max} - g_{ave}^* \leq \frac{\log |\mathcal{F}|}{\beta} \quad (18)$$

$$0 \leq g_{\max} - \bar{g}_{ave} \leq \frac{\log |\mathcal{F}|}{\beta} + \Lambda_{\max} \quad (19)$$

F. Insensitivity of Count-down Time Distribution

In general, the distribution of count-down time is not exponential. For example, the count-down time before the optimistic unchoking is 30 seconds in real BitTorrent implementations [1]. However, we have the following insensitivity result:

Theorem 2. *In implementations of perfect Markov chain, if we change the distribution of count-down time from exponential distribution to general distribution and keep the same mean of count-down time ($\frac{1}{\tau(\Delta-\delta)}$), then the stationary distribution of any configuration $f \in \mathcal{F}$ is still $p_f^*(\mathbf{g})$ in (12).*

G. Mixing Time of The Designed Markov Chain

In this subsection, we focus on the mixing time of perfect Markov chain.

Recall that \mathbf{p}^* (12) is the stationary distribution of the perfect Markov chain. Let $\mathbf{H}_t(f)$ denote the probability distribution of all states in \mathcal{F} at time t given that the initial state is f . We define mixing time of perfect Markov chain as follows:

$$t_{mix}(\epsilon) \triangleq \inf \left\{ t \geq 0 : \max_{f \in \mathcal{F}} d_{TV}(\mathbf{H}_t(f), \mathbf{p}^*) \leq \epsilon \right\} \quad (20)$$

where the total variance distance of any two probability distributions \mathbf{p}, \mathbf{p}' is defined as [9]:

$$d_{TV}(\mathbf{p}, \mathbf{p}') \triangleq \frac{1}{2} \sum_{f \in \mathcal{F}} |p_f - p'_f| \quad (21)$$

We have the following results.

TABLE II
DISTRIBUTION OF PEERS' UPLOAD BANDWIDTH

Upload (kbps)	256	378	512	768	1024	2048
Fraction (%)	40	5	5	5	5	40

Theorem 3. The mixing time $t_{mix}(\epsilon)$ for perfect Markov chain is bounded as follows:

(a) for general $\beta \in (0, \infty)$,

$$t_{mix}(\epsilon) \geq \frac{\exp(-\beta(g_{\max} - g_{\min}))}{2\tau \cdot \delta^{n-1}(\Delta - \delta)^n} \ln \frac{1}{2\epsilon} \quad (22)$$

$$t_{mix}(\epsilon) \leq \frac{2\delta^{n+1}(\Delta - \delta)^n}{\tau} \left(\frac{\Delta}{\delta}\right)^{2n} \exp(5\beta(g_{\max} - g_{\min})) \cdot \left[\ln \frac{1}{2\epsilon} + \frac{n}{2} \ln \left(\frac{\Delta}{\delta}\right) + \frac{1}{2}\beta(g_{\max} - g_{\min})\right] \quad (23)$$

where $g_{\max} = \max_{f \in \mathcal{F}} g_f$, $g_{\min} = \min_{f \in \mathcal{F}} g_f$ and $n = |V|$ denotes the number of peers in system.

(b) When

$$0 < \beta < \frac{1}{g_{\max} - g_{\min}} \ln \left[\left(1 + \frac{1}{\delta}\right) \left(1 + \frac{1}{\Delta - \delta - 1}\right) \right] \quad (24)$$

, we have a tighter upper bound:

$$t_{mix}(\epsilon) \leq \frac{\frac{1}{\tau(\Delta - \delta)} \cdot \ln \frac{n\delta}{\epsilon}}{1 - \left(1 - \frac{1}{\Delta - \delta}\right) \cdot \left(\frac{\delta}{\delta + 1}\right) \exp(\beta(g_{\max} - g_{\min}))} \quad (25)$$

Remarks: We discuss the trade-off between optimality gap (Theorem 1) and mixing time (Theorem 3). We consider two ends of this spectrum.

- As $\beta \rightarrow \infty$, the optimality gap approaches zero while the upper bound of mixing time scales with $\exp(\Omega(n))$ and approaches infinity (slow-mixing).
- As $\beta \rightarrow 0$, the optimality gap approaches infinity while the upper bound of mixing time scales with $O(\log(n))$ and remains limited (fast-mixing).

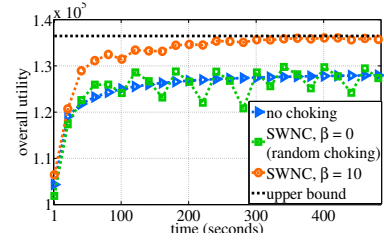
IV. EXPERIMENTAL RESULTS

A. Setup and Purpose

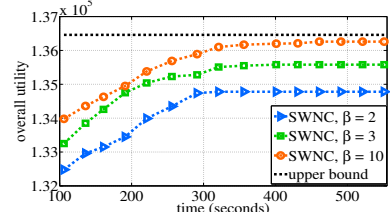
We chose the system parameters as follows: number of peers $|V| = 1000$, peers' upload neighbor size $\Delta = 20$, peers' upload degree bound $\delta = 4$. We also assume all peers' upload capacities follow the distribution shown in Table II and all peers' download capacities are equal to 512kbps.

Then we run our algorithm and compare its performance with random neighbor choking (essentially a special case of our algorithm at $\beta = 0$) and no choking. Corresponding results are shown in Fig. 2, where convergence time is defined as the time when the system's average utility does not change within 0.01% of its current value.

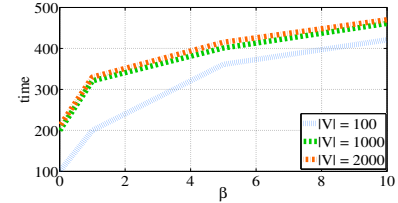
In Fig. 2 (a), we see that our algorithm achieves much better performance than other algorithms. In Fig. 2 (b), we show the impacts of β on system utility. As β increases, more utilities can be achieved. Further, even for a small value $\beta = 10$, our algorithm achieves close to optimal utilities with a small convergence time, i.e., less than 500 seconds. In Fig. 2 (c), we take a closer look of convergence time of the algorithm at different values of β and $|V|$ while keeping other parameters the same. We see that the convergence time increases almost linearly with β , and sublinear with the network size $|V|$. From the experiments, we see that theoretical bounds on utility gap



(a) Effectiveness of neighbor selection by comparing schemes of no choking, random choking and soft-worst-neighbor choking



(b) Impacts of β on system utilities



(c) Impacts of β and $|V|$ on convergence time

Fig. 2. Experiment Results

and mixing time can be loose, and in practice the performance is usually much better.

REFERENCES

- [1] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.
- [2] R. Xia and J. Muppala, "A survey of bittorrent performance," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 140–158, 2010.
- [3] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 203–216.
- [4] M. Chen, S. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *Proc. of IEEE INFOCOM 2010*, 2010, pp. 1–9.
- [5] N. Laoutaris, D. Carra, and P. Michiardi, "Uplink allocation beyond choke/unchoke," in *Proc. of ACM CoNEXT'08*, 2008.
- [6] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li, "Celerity: A low delay multiparty conferencing solution," in *Proc. of ACM Multimedia'11*, 2011.
- [7] M. Chiang, S. H. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.
- [8] S. Zhang, Z. Shao, and M. Chen, "Optimal distributed p2p streaming under node degree bounds," in *Proc. of IEEE ICNP 2010*, 2010, pp. 253–262.
- [9] D. Levin, Y. Peres, and E. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Society, 2009.
- [10] Z. Shao, H. Zhang, M. Chen, and K. Ramchandran, "Reverse-Engineering BitTorrent," *Technical Report*, 2011, available at <http://personal.ie.cuhk.edu.hk/~zyshao/bt.pdf>.
- [11] N. Hu, L. Li, Z. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: Algorithms, measurements, and implications," in *ACM SIGCOMM*, 2004.
- [12] H. Zhang, Z. Shao, M. Chen, and K. Ramchandran, "An Optimized Video-on-Demand System Based on Distributed Caching," *Technical Report*, 2011.