Gurtov, Andrei; Koskela, Joakim; Korzun, Dmitry

Cyclic ranking in single-resource peer-to-peer exchange

Published in:
Peer-to-Peer Networking and Applications

DOI:
10.1007/s12083-017-0578-0

Published: 01/01/2017

Document Version
Publisher's PDF, also known as Version of record

Peer-to-Peer Netw. Appl.
DOI 10.1007/s12083-017-0578-0

Cyclic ranking in single-resource peer-to-peer exchange

Andrei Gurtov[1] · **Joakim Koskela**[2] · **Dmitry Korzun**[3]

Received: 18 June 2015 / Accepted: 12 June 2017

Abstract Peer-to-peer (P2P) sharing systems use incentives for resource exchange to encourage cooperation and ensure fairness. In bilateral strategies, such as BitTorrent Tit-for-Tat or deficit-based FairTorrent, individual decisions of peers utilize direct observations. It may result in low performance and unfair treatment. In this paper, we study a novel exchange strategy that applies Cyclic Ranking (CR). In addition to direct observations, a peer utilizes provision cycles—a shared history of effective exchanges. The PageRank algorithm runs for the locally collected cycles and computes the numerical ranks to estimate the reputation. The CR strategy incrementally augments known incentive-aware strategies. For evaluation we implement CR-BitTorrent and CR-FairTorrent variants. Our simulation model captures the dependence on network bandwidth and the number of seeders as well as selfishness and stability of the participants. The initial experiments show improved fairness and download times, compared to the original BitTorrent and FairTorrent. The performance of selfish and unstable peers decreases by as much as 50%. The CR strategy suits well in environments where direct reciprocity has shown little effect. Contrasted to existing solutions, the CR strategy rewards longevity and stability of peers.

Keywords Peer-to-peer cooperation · Incentives · Reputation · Fairness · Provision cycles · Structural ranking · Measurements · Performance

1 Introduction

Single-resource exchange is a base component of P2P file-sharing networks. BitTorrent [1] and the multitude of its variants form one of the most demanded P2P protocol families [2]. Exchange in a BT-like system is based on bilateral incentives [3]: file downloading is possible in return for uploading to the same peer. A rational peer individually estimates reputation of other peers using direct observations of their provision. It is an approximated Tit-for-Tat strategy [1, 4]: each peer ranks its neighbors according to their direct provision. The BT strategy still suffers a certain level of unfairness and non-optimal performance. Although peers contribute upload bandwidth, there can be no appropriate return in the download performance.

Cyclic Ranking (CR) method [5] extends the direct observations with shared history in form of provision cycles. Research [3, 6–8] showed that such exchange paths do appear in P2P exchange due to its multilateral nature. Moreover, P2P reputation systems benefit from the use of structural ranking algorithms, such as EigenTrust [9] and Distributed PageRank [10], where connectivity properties of the network exchange influence the peer reputation.

This paper studies the applicability of the CR method for constructing effective incentives in BT-like systems. We contribute a novel CR strategy that provides multilateral incentives for cooperation. In addition to direct observation

✉ Andrei Gurtov
gurtov@acm.org

Joakim Koskela
jookos@gmail.com

Dmitry Korzun
dkorzun@cs.karelia.ru

[1] Linköping University, Linköping, Sweden

[2] Aalto University, Espoo, Finland

[3] Petrozavodsk State University, Petrozavodsk, Russia

Published online: 04 July 2017

of its neighbors, any peer can maintain locally a directed graph consisting of provision cycles. They are constructed by recommendations coming from reputable neighbors, hence reflecting effective exchange chains in the network. Links of the graph are weighted with local reputation of recommenders. The distributed construction allows the peers to disseminate their local reputation through the network. The peers run a variant of the PageRank algorithm on this graph, evaluating in the numerical ranks the reputation that captures exchange connectivity properties as well as direct observations.

The CR strategy has following properties:

1) CR can be implemented as an extension of a known BT-like strategy and incrementally deployed.
2) Provision cycles can be constructed individually for any peer by collective work where the participation has incentives and requires relatively small effort from every good participant.
3) CR ranks of peers provide proper incentives for cooperation, penalizing selfish and unstable peers and rewarding longevity and stability of peers.
4) CR ranks are resilient to false reports and collusion.

We implement several CR variants for extending the known strategies—BitTorrent and FairTorrent. For the evaluation we use an OMNeT-based simulator. We also observed that the performance of BitTorrent and FairTorrent depends on algorithmic implementation. Thus, one contribution of the paper is a detailed description of BitTorrent algorithms missing in related work.

The rest of the paper is organized as follows. Section 2 introduces background on P2P single-resource exchange with bilateral incentives. Section 3 contributes the CR strategy and explains its support of multilateral incentives. In Section 4, we specify the algorithms of several BitTorrent versions. Section 5 presents our simulation experiments and initial findings. Section 6 concludes the paper.

2 Single-resource exchange

File-sharing in BitTorrent (BT) [1] and in its numerous variants applies single-resource (single-torrent) peer-assisted exchange. The resource is the peer upload bandwidth in exchange of file pieces [3]. For a given file, BT exchange is based on a swarm of $N = N_S + N_L$ peers: N_S seeders are file owners providing file pieces to others "for free" and N_L leechers are interested in downloading the file. During the exchange, if a leecher u has downloaded some file pieces then u can upload them to other leechers. For absent pieces, u requests other peers of the swarm. Eventually, every leecher completes the file download.

The demand of every leecher is matched with available supply at other peers. They are motivated to contribute the bandwidth in a bilateral manner: download is possible in return for upload to the same peer. The objective is reducing the download performance for free-riders and other "parasite" participants. The fairness is supported on the microscopic view [11]: u uploads to v similarly to u has downloaded from v, based on u's local estimations for v, for all leechers u and v. An individual algorithm a leecher uses in BT exchange is called a client. There are many implementations for practical use (e.g., Vuze (Azureus), μTorrent) and for research (e.g., FairTorrent, PropShape, BitThief, BitTyrant).

A client divides runtime onto rounds, which are not synchronized among peers. Typical BT clients use 10–20 s rounds. FairTorrent [12] applies "more instant" strategy where every upload action (sending a data block) starts the next round. The similarity of upload and download is estimated locally using direct observations from several last rounds.

Each u tries to maintain n_u connections (TCP) with other peers in the swarm (e.g., $n_u = 50$ in Vuze, $n_u = 80$ in original BT). Local routing table T_u stores all open connections ($n_u = |T_u|$). It represents the neighborhood that u knows. The control traffic required for data exchange is minimal: each peer transmits messages indicating the data blocks it currently possess and messages signaling their interest in the blocks of other peers.

A leecher u selects n_{bst} neighbors v of the highest download rates a_v (v's provision history). Besides, u selects n_{uch} peers for optimistic unchoking. It aims at finding new neighbors in the swarm that would offer good download bandwidth. The original BT client unchokes peers randomly. Some popular BT client implementations (e.g., Azureus), on the other hand, make a weighted random choice that takes into account the exchange history with a peer.

The $n_{bst} + n_{uch}$ neighbors are active, and u uploads only to them during the current round. For downloading u may exploit all peers and consume as much as possible from them.

The above strategy is described in terms of ranks. Completing round t, a peer u computes ranks $r_v(t + 1)$ that numerically estimate the observed v's provision. In round $t+1$, provision of u to v depends on the rank-based arrangement of neighbors. Table 1 shows ranks for BitTorrent (BT), PropShare (PS), and FairTorrent (FT). For new joining peers the direct rank is zero or can be set to a fixed small value.

Table 1 Local direct ranks at a leecher u to arrange its neighbors

BT [1]	$r_v = 1/(n_{\text{bst}} + n_{\text{uch}})$ if v is one of the n_{bst} best neighbors (by a_v) or v is one of the n_{uch} unchoked peers; $r_v = 0$ otherwise.
PS [4]	$r_v = a_v / \sum_w a_w$, where the sum is over all N peers excluding u.
FT [12]	Rank is deficit $r_v = a_v - b_v$, where a_v and b_v are v's download to u and upload from u, respectively.
BTyr [13]	Rank is share ratio $r_v = a_v/b_v$.

The reference BT client with equal ranks for the top downloaders (active set) is not used in recent implementations.

The original BT strategy suffers from a certain level of unfairness [12]. Although peers contribute upload bandwidth, there can be no appropriate return in the download performance. One of the reasons is optimistic unchoking. Even making $n_{\text{uch}} = 1$ does not properly restrain selfish strategies of other peers (e.g., BitTyrant) [13]. Another reason is the scarce granularity of contributing neighbors: small n_{bst} (typically, $n_{\text{bst}} = 4$) and treating all active neighbors equally in uploading. The PS strategy attempts to solve this issue by considering all possible neighbors (up to N) and splitting the peer's upload bandwidth in proportion to the contribution received in the previous rounds.

Peer behavior is subject to frequent changes, random or strategic. The long-round strategies of BT and PS have slow reaction (typical round is 30 s). Selfish peers can earn points in t and become free-riders in $t+1$. In contrast, the FT strategy does not fix round length: u changes v's rank (deficit counter) whenever observing any download/upload activity of v. Thus, u reacts immediately to changes in v's behavior. On the other hand, changes can be occasional, e.g., due to intermittent network connectivity. The FT strategy is short-term, not focusing on longevity and stability of the participation.

3 Cyclic ranking

Bilateral incentives are less efficient than multilateral [3, 14]. Direct observations provide small coverage, so discovery of good participants is slow. If peers share their observations, then a peer can locally approximate the global exchange activity for better reputation estimation. We apply the cyclic ranking (CR) ranking method of [5, Ch. 10] and develop a CR strategy for improving fairness and performance in BT-like systems.

A BT swarm is an almost full-connected network—each peer knows most of others. Considering only active upload links for each peer u, the instant exchange topology is a directed graph. A link $u \to v$ has weight c_{uv}, e.g., deficit counter or download/upload ratio. Given this global graph, a ranking algorithm [15] computes ranks estimating the reputation of peers. The computation is either 1) centralized—a non-scalable solution for P2P, 2) distributed [10]—vulnerable to false reports, or 3) local—the global knowledge assumption is unrealistic [5].

3.1 Provision cycles

Let $G = (N, L)$ be a directed graph that describes the global exchange topology. It has no parallel links and self-loops.

A provision cycle is an exchange chain in the swarm,

$$u \to w_1 \to w_2 \to \ldots \to w_i \to \ldots \to w_l \to u. \tag{1}$$

Every link $w_i \to w_{i+1}$ is "good" in the bilateral exchange, i.e., w_{i+1} is a stable productive neighbor of w_i. Cycles Eq. 1 exist in P2P economies [6, 7, 16]; they are short and have reasonable interpretation.

A BT-like strategy uses 2-hop cycles $u \to v \to u$, which are easily identifiable by direct observations.

3.2 Subjective cyclic graph

Let a node u create its own current local view of the exchange topology, as in the BarterCast reputation mechanism [17]. Denote[1] $G_u = (N_u, L_u; C)$ the weighted u's subjective graph, which is a personalized and partial view of G. In accordance with the basic CR method [5, Ch. 10], G_u consists of provision cycles Eq. 1. Further let us call G_u the u's cyclic graph.

Consider how u can update its cyclic graph G_u based on new cycles coming as recommendations from neighbors. Let v recommend m_v cycles

$$v \to w_{j1} \to \ldots \to w_{jl_j} \to v, \quad j = 1, \ldots, m_v, \tag{2}$$

where $w_{jl} \neq u$ for $l = 1, \ldots, l_j$. Denote $w_{j0} = u$ and $w_{j,l_j+1} = v$.

3.2.1 Topology update

Construct $1 + m_v$ cycles

$$u \to v \to u,$$
$$u \to w_{j1} \to \ldots \to w_{jl_j} \to v \to u, \quad j = 1, \ldots, m_v.$$

[1] For the simplicity the notation does not show the dependence of C on u.

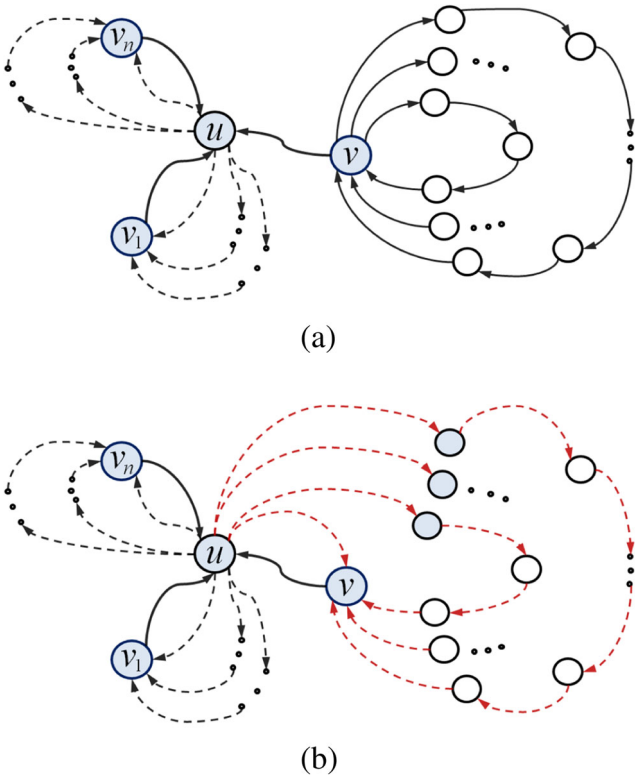


Fig. 1 Cyclic graph construction: **a** v recommends its cycles to u; **b** u transforms the cycles and embeds them into its cyclic graph

(3)

The idea is illustrated in Fig. 1. The first cycle in Eq. 3 is the default 2-hop cycle. Peers w_{jl} expand the u's local knowledge. (Though some of them can be already in N_u.)

Update Eq. 3 can be applied even if $m_v = 0$ (no recommendation from v). All cycles

$$u \to v \to u \quad \forall\, v \in T_u \tag{4}$$

form the minimal cyclic graph that u can construct. Graph Eq. 4 can be used for the initialization.

A cycle in Eq. 2 reflects a good provision chain observable in the exchange topology (Fig. 1a). Update Eq. 3 introduces new good provision cycles to the u-aware multilateral exchange, and u becomes their head (Fig. 1b).

3.2.2 Reputation-aware link weights

Any indirect information from v is reputed by u in respect to v's provision. The local direct rank r_v provides the reputation bounds (see Table 1).

Update Eq. 3 defines $1+m_v$ cyclic flows, where $v \to u$ is the only common link. Let this u-incoming link have weight r_v. All the $m_v + 1$ outgoing links and their succeeding links are of uniform weights $r_v/(1 + m_v)$. This way is similar to flow-based reputation schemes, and leads to the circulation flow of r_v units started from u.

Algorithm 1 formulates our cyclic graph construction. Any leecher u can perform it individually. The construction is personalized: u can make own decision on how to combine previous (rounds $1, 2, \ldots, t-1$) and recent (round t) observations.

Algorithm 1 Cyclic graph construction at a leecher u in every round $t = 1, 2, \ldots$

Require: Neighbors $N_u(t)$ and their normalized recent local direct ranks $\sum_{v \in N_u(t)} r_v^* = 1$.
Ensure: Cyclic graph $G_u(t)$ with link weights C

1: let $G_u(t)$ consist of selected cycles from $G_u(t-1)$
2: **for all** $v \in N_u(t)$ **do**
3: $r_v(t) = \alpha_r r_v(t-1) + (1-\alpha_r) r_v^*$
4: add $u \to v \to u$ to $G_u(t)$
5: $c_{uv} \mathrel{+}= r_v(t)$, $c_{vu} \mathrel{+}= r_v(t)$
6: **if** v is a good provider **then**
7: let u receive cycles (2) from v
8: **for all** cycles $j = 1, \ldots, m_v$ **do**
9: add $u \to w_{j1} \to \ldots \to w_{jl_j} \to v \to u$ to $G_u(t)$
10: $c_{w_{jl}, w_{j,l+1}} \mathrel{+}= \frac{1}{1+m_v} r_v(t)$ for $l = 0, 1, \ldots, l_j+1$
11: **end for**
12: $c_{uv} \mathrel{-}= \frac{m_v}{1+m_v} r_v(t)$
13: **end if**
14: **end for**

On Step 1 the leecher u inherits selectively cycles found previously ($G_u(0)$ is empty). The selection problem is beyond the focus of this paper. In our experiments, u initializes $\mathcal{C}_u(t) = \varnothing$.

The loop for $v \in N_u(t)$ considers possible cycles coming from the neighbors.

Step 3 evaluates initial node ranks $r_v(t)$ using direct local ranks r_v^* (see Table 1) and exponential moving average with degree $0 \le \alpha_r < 1$. It makes a tradeoff between previous and recent behavior.[2] If $\alpha_r = 0$ then the initial rank and direct rank coincide, i.e., the case of BT-like algorithm. If $\alpha_r > 0$ then rounds of low provision reduce $r_v(t)$, even despite of high recent provision in t. Similarly, a new neighbor v can achieve high $r_v(t)$ if v was a good provider for u in few latest rounds $\tau < t$. In our simulation experiments, $\alpha_r = 0.5$.

On Step 4, all default 2-hop cycles are added[3]. Then the link weights are updated with the initial v's rank on Step 5. They can be changed on further iteration of the cycle (Steps 2–14) due to recommendations from other neighbors.

[2] If no previous ranking is available for v then let $r_v(t-1) = 0$.

[3] By default, $c_{vw} = 0$ for any new link $v \to w$ added to G_u.

Step 6 restricts the consideration of u to its best neighbors, e.g., $r_v \geq \bar{r}$ for a reasonable threshold $\bar{r}$. Only good providers can recommend cycles to u and make u generous for the peers that feed v. In the basic case, v recommends its best neighbors w, resulting in 3-hop cycles $u \to w \to v \to u$ at u. In our experiments we consider a simple case when any leecher v recommends all cycles it knows. Steps 7–12 implement the topology and link weight update in G_u based on incoming recommendations.

Importantly that v must donate some of its rank r_v to the link weight for other nodes in G_u. Thus u will upload more to those peers (link weight increment on Step 10). The sum weight increment to all m_v links $u \to w_{j1}$ equals $r_v(t)m_v/(1+m_v)$, which is subtracted from the weight of $u \to v$ on Step 12. Although this donation reduces the direct upload from u to v, the latter expects provision benefit from the recommended peers. Moreover, v-incoming links can achieve additional weight due to recommendation from other u's neighbors if v has showed its good provision for them. Taking into account all $v \in N_v$, the link weight is made high if 1) many neighbors v recommend it, 2) recommenders are good (high r_v), 3) recommenders provide a moderate set of cycles (low m_v).

3.3 Cyclic PageRank

Cyclic graph G_u represents knowledge on the u-aware exchange topology. The graph can be used for personalized local ranking (in respect to u) using known graph-based (structural) algorithms such as PageRank [15, 18].

PageRank was introduced for ranking web pages. A link from a page to another is an endorsement indicating the quality of the latter page. The background model is a random walk with probability of step $u \to v$ relative to link weight c_{vw}:

$$\bar{c}_{vw} = c_{vw} \Big/ \sum_{w'} c_{vw'},$$

forming the relative weight matrix $\bar{C}$. At each step a node v selects with probability $\bar{c}_{vw}$ a link $v \to w$ to follow. Assuming the steady state exists, the PageRank value p_v is the probability that the random walk is in v. The computation is iterative:[4]

$$p_v^{(i+1)} = \sum_{w \to v} \bar{c}_{wv} p_w^{(i)}, \tag{5}$$

starting from initial values $p_v^{(0)}$ and converging to p_v.

Similarly to web pages, a link $v \to w$ in G_u is considered as indication of the quality of w for v. The idea was previously exploited in distributed PageRank [10] and EigenTrust [9], where c_{vw} estimates u's trust for $v \to w$. In contrast, Cyclic PageRank is not based on the global exchange topology graph. The global graph assumption can be inadequate for unique link weighting: the same link $v \to w$ can easily appear good in respect to u_1 and bad in respect to u_2. That is, a rational u is interested in personalized knowledge (G_u).

For given G_u let us further call p_v the cyclic rank values (CR values), which are a non-negative real solution to

$$p = \bar{C}^{\mathsf{T}} p \tag{6}$$

normalized with $\sum_{v \in N_u} p_v = 1$. Since G_u is always strongly connected, no damping factor is needed for ensuring the steady state existence and convergence in Eq. 5. Rank p_v becomes high when v appears in many cycles (many $w \to v$) with high link weights ($\bar{c}_{wv}$) and high-rank predecessors (p_w).

3.4 Analysis

Let us prove that in case of no recommendations Algorithm 1 provides such ranks that coincide with direct ranks r_v.

Theorem 1 *For minimal cyclic graph G_u the equality $p_v = r_v$ holds.*

Proof Any minimal cyclic graph is defined by Eq. 4 for a given weight matrix C. The probability of being in some $v \in T_u$ during random walking in G_u is simply determined with the normalized solution to Eq. 6:

$$p_v = c_{uv} \Big/ \sum_{w \in T_u} c_{uw}.$$

Any weight c_{uv} is initially equal to the local direct rank (see Table 1), and p_v coincides with the normalized r_v. □

A rational peer v is interested in that many peers recommend it in their cycles. Such recommendations propagate and increase the rank p_v at v's neighbors. This intuition substantiates the incentives for peers to share cycles.

By Algorithm 1, a neighbor v always becomes the last node in the cycles. This property reduces p_v if v's cycles consist of bad providers, so sacrificing v itself in the case of false reporting. Attempts for recommending many cycles (possibly consisting of sibling peers) lead to low link weights, since the direct provision/consumption ratio r_v is partitioned among all cycles from v. Consequently, v is interested to recommend a moderate set of good cycles.

3.5 CR-extended BT-like exchange

There are different options of integrating CR into BT exchange, depending on how cycles are accumulated and

[4] As we shall see later, Cyclic PageRank does not require a damping factor.

how much they can affect the behavior of peers. By leveraging shared history in BT-like systems, the peers make more long-sighted decisions, possibly at the expense of immediate rewards. We explore different options by evaluating two strategies: CR-BT and CR-FT.

CR-BT represents a conservative approach, utilizing CR in a very unobtrusive manner. It extends the vanilla BitTorrent algorithm by modifying only two choices made by peers (which are normally given less attention to): optimistic unchoking and peer selection.

BT optimistic unchoking aims at discovering better peers by choosing n_{uch} random peers to unchoke at every round. CR-BT modifies this stochastic mechanism using the probability distribution relative to the peer CR ranks. The probability of v being unchoked by u becomes

$$\pi_v = p_v / \sum_w p_w. \tag{7}$$

Note that Eq. 7 cannot use the local ranks r_v from Table 1, since r_v are available for active peers only. In contrast, cyclic graph $\mathcal{C}_u$ contains peers that are observed directly or from (indirect) recommendations of good neighbors.

In BT selection, u chooses in the whole swarm a few peers to which an active connection is maintained. The CR-BT strategy selects the peers based on their CR ranks p_v. This bolsters the effect of the CR-BT unchoke, as the peers connecting to u are also more likely to esteem u (in terms of CR), and thus unchoke.

CR-FT is an extension of FairTorrent, capturing its ultra reactive choke*less* model. An FT peer chooses the next packet to transmit based on the instant deficits (local ranks r_v) of its neighbors. A CR-FT peer prioritizes the CR ranking p_v over the deficit and chooses v^* of the highest CR rank among all peers that u knows:

$$v^* = \operatorname{argmax}_v \{p_v\}. \tag{8}$$

The CR-FT strategy relies therefore on the shared history (if u accumulated it already), rewarding peers that have contributed to the network as a whole, not only recently to u (its current neighbors).

4 Algorithms / methodology

The purpose of the simulations is to discover the behavior of the proposed schemes compared to existing ones, and asses their benefits and suitability for different scenarios. Specifically, we look at the base performance of the different algorithms in terms of completion times for individual peers in order to see how the algorithms change the incentives for peers to behave in different ways.

4.1 Simulator description

We observed that behavior of different BitTorrent versions depends a lot on the details of algorithms for peer selection and bandwidth allocation which are often hidden in the implementation details. Therefore, we see it as an important contribution to list all key algorithms explicitly to facilitate reproductivity of experiments and comparison with future BitTorrent enhancements.

We implemented those algorithms using an OMNeT++ simulator framework-based BIT-SIM simulator. The BT swarm simulator consists of a *Tracker* and a group of peers. The peers were assigned one of the following roles according to each simulator's parameters which are described below: a seeder, leecher, selfish or unstable.

All connections between peers and the tracker are constructed using identical bandwidth-limited data links, creating a homogeneous network. To enforce a strict total uplink bandwidth limit independently of the number of peer connections, the simulator uses a queuing system for packet transmission; each peer would transmit using only a single data link at a time, ensuring that the total uplink bandwidth equals that of the individual links.

The data processing events were completed without artificially introduced delay (all processing is instantaneous in the simulation's timeframe).

4.2 Client behavior

The BitTorrent clients' behavior can be divided into the following phases: 1) a periodic timer, 2) when receiving data, 3) when transmitting data, 4) when receiving a new peer connection.

All BT leecher peers' timers follow the following pattern. During initialization, each peer would be assigned a unique identifier and register with the tracker. After that, and every 30 minutes, peers would request a list p_{pot} of 100 active peers from the tracker, which is constructed by randomly selecting peers from all known to the tracker p_{all} (Algorithm 2). After acquiring p_{pot}, and every 5 minutes, peers would discard peers already connected, creating a list of potential new peers p_{new}. From this list, peers would be selected using the function $SelectPeer()$, and a connection would be attempted to be established until the total number of peer connections is 50 or p_{new} is empty.

Algorithm 2 Connecting to new peers

let $\mathcal{P}_{new}$ consist of new peers
while $totalConnections < 50$ and $\mathcal{P}_{new}.length > 0$ **do**
  $p = SelectPeer(\mathcal{P}_{new})$
  if $Connect(p)$ succeeds **then**
    $totalConnections \leftarrow totalConnections + 1$
  end if
  $\mathcal{P}_{new}.Remove(p)$
end while

In BT and FT, the $SelectPeer()$ will uniformly select a random peer. In CR-BT and CR-FT, the peer will be randomly selected using its cyclic rank value as weight (Algorithm 3).

Algorithm 3 $SelectPeer()$ for CR-BT, CR-FT

Require: $\mathcal{P}_{new}.length > 0$
$s = \sum_{p \in \mathcal{P}_{new}} Rank(p)$
if $s > 0$ **then**
  $r = Uniform(0..s)$
  for all p in $\mathcal{P}_{new}$ **do**
    if $Rank(p) \geq r$ **then**
      $r = r - Rank(p)$
    else
      $p_{selected} = p$
    end if
  end for
else
  $p_{selected} = Uniform(\mathcal{P}_{new})$
end if
return $p_{selected}$

Selecting which data requests to respond to is done either through *choking* (ignoring requests) or recipient selection. FT and CR-FT does not use choking, instead they accept all requests, but respond to these selectively. BT and CR-BT use choking, but respond to all requests they accept. In order to select which peers to unchoke, or the next request to respond to, both the number of bytes received and transmitted is recorded for each peer.

Selection of the peers to unchoke in BT and CR-BT is done every 10 seconds (Algorithm 4). BT uses the peers' performance (bytes received from the peer) during the last 10 seconds as criteria, while CR-BT uses the peers' ranks. Each round, four peers are unchoked, with a fifth chosen randomly (BT) or based on the ranks (CR-BT) each 30 seconds (using $SelectPeer()$ defined earlier).

Algorithm 4 Choke- selection in BT, CR-BT

Require: $\mathcal{P}_{all}$ contains all connected peers

$leftToUnchoke \leftarrow 4$
let $\mathcal{P}_{choked}$ be empty
$\mathcal{P}_{sorted} = SortPeersByRecv(\mathcal{P}_{all})$
for all p in $\mathcal{P}_{sorted}$ **do**
  if $leftToUnchoke > 0$ and $IsInterested(p)$ **then**
    $Unchoke(p)$
    $leftToUnchoke \leftarrow leftToUnchoke - 1$
  else
    $Choke(p)$
    if $IsInterested(p)$ **then**
      $\mathcal{P}_{choked} \leftarrow p$
    end if
  end if
end for
if 30 seconds passed and $\mathcal{P}_{choked}.length > 0$ **then**
  $p = SelectPeer(\mathcal{P}_{choked})$
  $Unchoke(p)$
end if
set $p.recv = 0 \forall p \in \mathcal{P}$

In contrast, FT and CR-FT unchoke all peers that express interest. In BT and CR-BT, requests are responded to in the order they arrive.

The FT algorithm uses the total deficit for each peer, i.e., the difference between the number of bytes sent and received over the lifespan of the connection, to select the next request to respond to (Algorithm 5). The first request from the peer with the lowest deficit is chosen.

Algorithm 5 *SelectNextRequest($\mathcal{Q}$)* for FT

Require: $\mathcal{P}_{DF}$ contains all peers, sorted by deficit $DF = peer.sent - peer.recv$, in ascending order
for all p in $\mathcal{P}_{DF}$ **do**
  if $IsConnected(p)$ **then**
    for all r in $\mathcal{Q}$ **do**
      if $r.peer == p$ **then**
        return r
      end if
    end for
  end if
end for

CR-FT chooses the first request by the peer with the highest rank value (Algorithm 6).

Algorithm 6 *SelectNextRequest($\mathcal{Q}$)* for CR-FT

Require: $\mathcal{P}_{Rank}$ contains all peers, sorted by rank in ascending order

```
for all p in P_Rank do
  if IsConnected(p) then
    for all r in Q do
      if r.peer == p then
        return r
      end if
    end for
  end if
end for
```

4.3 Malicious- and misbehaving nodes

The selfish nodes operate identically to normal leechers of the selected algorithm (including connection establishment, when requesting pieces, the choke mechanism), except when receiving a piece request from a peer. The requests are simply discarded without informing the requestor.

Unstable nodes also operate identically to normal leechers, but initiate periods when no packets (of any kind) are sent or received, simulating a client who either logs off unexpectedly for some time, or has a bad network connection (Algorithms 7). The packets, which include piece requests, data blocks and control packets, are discarded completely during the periods of instability. The instability is triggered by a timer which is set to fire after a time $\mathcal{T}\int = uniform(\mathcal{T}\int_{min}, \mathcal{T}\int_{max})$, following the last period of instability. The instability lasts for a time $\mathcal{T}\sqcap = uniform(\mathcal{T}\sqcap_{min}, \mathcal{T}\sqcap_{max})$.

Algorithm 7 *InstabilityTrigger(p)*

Require: p has been configured as an unstable peer

```
if p.CurrentlyUnstable then
  delay = uniform(T∫_min, T∫_max)
  p.CurrentlyUnstable = False
  TriggerAfter(InstabilityTrigger, delay)
  p.RequestPieces()
else
  delay = uniform(T⊓_min, T⊓_max)
  p.Unstability ← p.Unstability + delay
  p.CurrentlyUnstable = True
  TriggerAfter(InstabilityTrigger, delay)
end if
```

After the simulation, the completion times for unstable nodes are calculated using only the periods of stability. This provides us with comparative values that highlight only the effect of the unstable behavior.

4.4 Round-robin seeder

Seeders use the choking mechanism, as BT and CR-BT, to select which peers are provided data. Up to 4 peers are unchoked each 10 seconds in a round-robin manner (Algorithm 8).

Algorithm 8 Round-robin seeder choke selection

Require: $\mathcal{P}$ contains all peers

```
let P_unchoked be empty
for all p in P do
  if IsConnected(p) then
    if P_unchoked.size < 4 and IsInterested(p)
    then
      Unchoke(p)
      P.Remove(p)
      P_unchoked ← p
    else
      Choke(p)
    end if
  end if
end for
{Add the unchoked peers to the end of P}
for all p in P_unchoked do
P.Append(p)
end for
```

5 Simulation

We evaluated the CR strategies using a simulator developed with the OMNeT++ tool [19]. Different set-ups allow comparing the performance to regular BT and FT strategies when the proportions of different types of peers are varied. We also simulated mixed-strategy networks in order to determine the benefits in gradual deployment.

5.1 Model

The model focuses on evolution of an N-size swarm. Given a swarm, its primary client strategy is fixed. The file size is F. Let all peers be classified into types $i = 0, 1, \ldots, n$, where type $i = 0$ represents seeders and types $i > 0$ represent leechers of different behavior. The type ordering is partial in terms of the selfishness level, i.e., for some $i < j$ we cannot assert that type j is more selfish than i.

Table 2 Basic types of peer behavior

Name	Description
Seed	It has all data and altruistically provides them to others in a round-robin fashion.
Good	It follows the primary client algorithm (strategy), i.e., BT, CR-BT, FT, or CR-FT.
Unstable	It alternates good behavior and inactivity. The periods follow exponential distributions with means $1/\lambda_{\text{good}}$ and $1/\lambda_{\text{inact}}$, which are typically 5..10 minutes long.
Lazy free-riding	It does not upload to other peers and does not advertise having data. Appears always as a newly arrived peer to others.
Deceptive free-riding	Appears as a normal good peer (wrt. choking, data advertising), except that it never uploads the data it is requested of. Disguises itself as a peer with very low upload capability.
Aggressive free-riding	As a lazy free-rider, but u seeks to connect to more peers (up to $n_u = 500$) increasing its exposure to seeders and optimistic unchoke. Aggressively polls the tracker for new peers. For the BT strategy it coincides with BitThief client [20].

The basic set of types for our experiments is shown in Table 2 ($n = 5$). It can be easily extended, e.g., BitTyrant client [13] allows benefit (if possible) by deviating strategically from the strict bilateral balance.

Let N_i be the number of peers of type i in the swarm, $N = \sum_{i=0}^{n} N_i$. At the start of any simulation the population size parameters N and $(N_i)_{i=0}^{n}$ are fixed. Among N peers the n types are assigned at random.

Throughout the simulation, the parameters remain constant. Any seeder stays in the swarm forever. After completing a download, the leecher is replaced by a new peer of the same type. Therefore the model reflects a stable regime of exchange, and we focus on evolutionary long-term characteristics of fairness and performance.

The basic performance metrics are summarized in Table 3. Clearly $B_i = L_i/T_i$. Applying Little's law, we yield $N_i = \lambda_i T_i$, where λ_i is the departure rate[5] of leechers of type i. Contrasting L_i to F estimates the fairness.

The model supports sequential event-driven sampling. An event is a departure of next leecher u due to download completion. If u is of type i then the next sample for every metric ($T_i(u)$, $L_i(u)$, and $B_i(u)$) is calculated.

Comparison of swarms with different primary strategies evaluates improvements in the fairness and performance. Intuitively, an efficient strategy makes 1) T low for good leechers and higher for leechers with more selfish behavior, 2) L comparable with or less than F for good leechers.

[5]The departure and arrival rates are equal since N_i is fixed.

Table 3 Performance metrics

Symbol	Description
T_i	The average download completion time (sec) for a leecher of type $i > 0$, i.e., leecher lifetime. For a given leecher u we also use $T_i(u)$ or $T(u)$.
L_i	The total amount (KB) that a leecher of type $i > 0$ has uploaded on average being in the swarm. For a given leecher u we also use $L_i(u)$ or $L(u)$.
B_i	The average upload throughput (KB/sec) that a peer of type i contributes during the exchange. For a given peer u we also use $B_i(u)$ or $B(u)$.

5.2 Experiments

Our BitTorrent simulator is based on the work described in [21]. The implementation was ported to OMNeT++ 4.2.2, adding additional logging capabilities, configurable peer and seeder behavior, and the latest versions of BT algorithms (including FairTorrent).

The experiments consist of more than 600 individual simulations, each implements a swarm for sharing a 650MB torrent file.

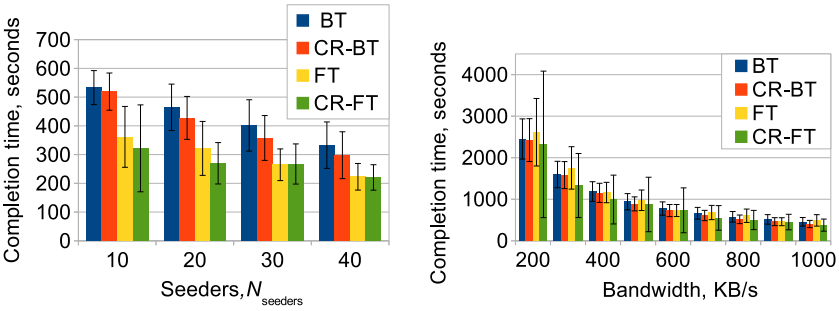
Each simulation was run five times, with different role assignments, using a homogeneous network (bandwidth and peer capabilities). The bandwidth of each peer was limited only on outgoing (provisioning) traffic, and had a set delay of 300ms.

The cycle distribution for the CR strategies is implemented using custom BitTorrent extension packets. The cycles (and ranks) were updated periodically every 60 seconds, with the cycle length limited to 5 nodes.

5.3 Analysis

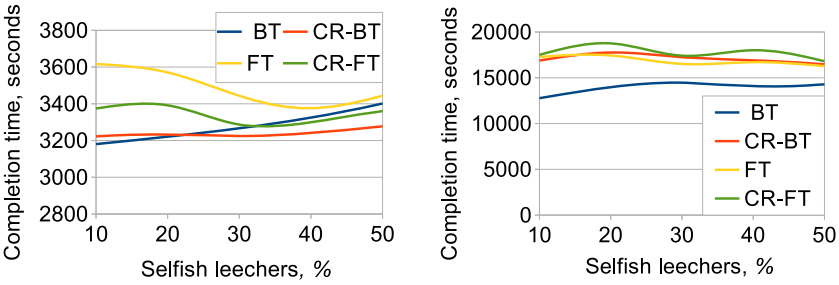
We measured the base performance of the CR strategy in networks without selfish or unstable peers. Figure 2 shows that both CR-BT and CR-FT perform as well or better than their counterparts with regards to the average download completion time for leechers. The improvement is roughly 10% between BT and CR-BT throughout the simulations. The impact of CR-FT was more diverse, showing 10..24% shorter times in restricted networks (low bandwidth and N_{seeder}), while performing as well as normal FT in others. On the other hand, the CR strategies increase the variation.

Our simulations showed that CR decreases the average download time for good leechers by 2.6% (Fig. 3) when there are selfish peers. Nevertheless, CR has no substantial impact on discouraging specifically selfish behavior compared to FT, as the slightly shorter completion times can be attributed to the general swarm efficiency improvement seen earlier. Compared to BT, the CR strategy does show



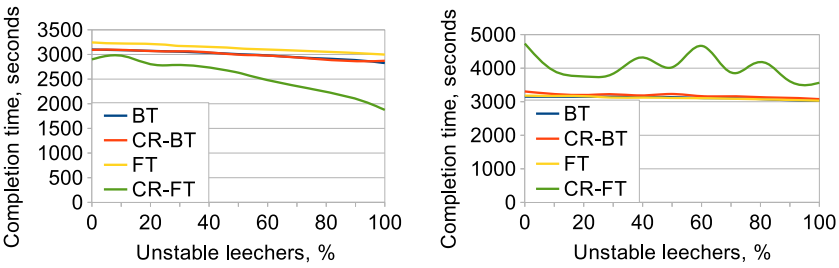
(a) Bandwidth 2MB/s (b) $N_{\text{seeder}} = 30$

Fig. 2 Average completion times in dependence on the share of seeders and bandwidth for $N = 100$



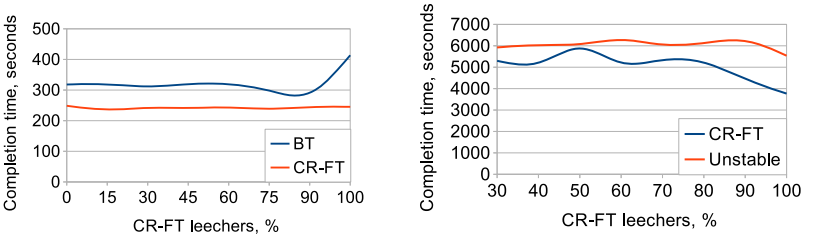
(a) Good nodes (b) Selfish nodes

Fig. 3 Average completion times in presence of selfish peers: bandwidth 200KB/s, $N_{\text{seeder}} = 10$, $N = 100$



(a) Good nodes (b) Unstable nodes

Fig. 4 Average completion times in presence of unstable peers. The periods of inactivity are subtracted. Bandwidth 200KB/s, $N_{\text{seeder}} = 3$, $N = 30$



(a) Bandwidth 2MB/s, $N_{\text{seeder}} = 10$, $N = 100$ (b) Bandwidth 200KB/s, $N_{\text{unstable}} = 30$, $N_{\text{seeder}} = 3$, $N = 30$

Fig. 5 Average completion times depending on the share of CR-FT leechers

noticeable improvement, increasing the completion times for selfish peers by 15..32%.

The most interesting results were observed when introducing unstable peers. This had little effect on the performance of BT, CR-BT and FT; the figures for unstable peers were indistinguishable from good peers (when subtracting the periods of inactivity). In contrast, CR-FT shows very strong reaction, especially in low-bandwidth networks (Fig. 4). The completion time for unstable peers increases by 32% on average, in some cases by as much as 50%. Stable peers experience shorter times as the share of unstable peers is increasing.

The increase for unstable peers is explained by the CR ranks. They decrease during inactivity, leaving the peers less likely of being served. The stable peers having stable ranks naturally benefit from this drop. As the number of unstable peers increases, the number of better ranked peers competing for the same resource decreases, leading to better service.

Finally, we experimented with mixed swarms consisting of leechers following both BT and the CR strategies. Overall, in a network with only good leechers, the completion times remained stable even though the share of CR-nodes varied (Fig. 5a). This suggests that the CR strategy (especially CR-FT) is beneficial even when sparsely deployed. However, in the presence of unstable nodes (Fig. 5b), the performance of CR-FT peers depends on the number of leechers implementing the strategy. The completion times for CR-FT leechers increase as the share of them decreases, suggesting that a wide deployment is required to efficiently discourage unstable behavior.

6 Conclusion

Bilateral single-resource exchange in P2P sharing systems, which is based on direct observations, can be effectively extended with additional indirect information that comes from multilateral exchanges. This information is represented by provision cycles, which naturally exist in such systems as previous studies showed. We applied the Cyclic Ranking method and proposed an advanced peer strategy (CR strategy) for BitTorrent-like systems. The CR strategy can be implemented there as an extension allowing incremental deployment. These properties are confirmed by implementation for BitTorrent and FairTorrent.

Our initial experiments showed that rational peers have certain incentives for using the CR strategy. In a network with selfish peers, CR slightly reduces the average download time for good leechers and significantly punishes selfish peers. In presence of unstable peers, CR is the only strategy that punishes the instability and encourages the stability. The overhead of cycle collection was estimated at few percent of bandwidth, thus not affecting the download rate noticeably. The limitations of our work include a simple churn model using the unstable peers and homogeneous network topology. In future work, we plan to extend the model to include multiple swarms per a single client.

Acknowledgments The work of D.Korzun is financially supported by the Ministry of Education and Science of Russia within project no. 2.5124.2017/8.9 of the basic part of state research assignment for 2017–2019. Andrei Gurtov was supported by the Center for Industrial Information Technology (CENIIT).

References

1. Cohen B (2003) Incentives build robustness in bittorrent. In: Proceedings of the 1st workshop on economics of peer-to-peer systems
2. Dhungel P, Wu D, Ross KW (2009) Measurement and mitigation of BitTorrent leecher attacks. Comput Commun 32(17):1852–1861. [Online]. Available: doi:10.1016/j.comcom.2009.07.006
3. Aperjis C, Freedman MJ, Johari R (2011) Bilateral and multilateral exchanges for peer-assisted content distribution. IEEE/ACM Trans Netw 19:1290–1303
4. Levin D, LaCurts K, Spring N, Bhattacharjee B (2008) BitTorrent Is an auction: analyzing and improving BitTorrent's incentives. SIGCOMM Comput Commun Rev 38:243–254
5. Korzun D, Gurtov A (2013) Structured peer-to-peer systems: fundamentals of hierarchical organization, routing, scaling and security. Springer
6. Anagnostakis KG, Greenwald MB (2004) Exchange-based incentive mechanisms for peer-to-peer file sharing. In: Proceedings of the 24th international conference on distributed computing systems (ICDCS'04), ser. ICDCS '04. IEEE Computer Society, pp 524–533
7. Menasché DS, Massoulié L, Towsley D (2010) Reciprocity and barter in peer-to-peer systems. In: Proceedings of IEEE INFOCOM'10. IEEE, pp 1505–1513
8. Liu Z, Hu H, Liu Y, Ross KW, Wang Y, Mobius M (2010) P2P trading in social networks: the value of staying connected. In: Proceedings of IEEE INFOCOM'10. IEEE, pp 2489–2497
9. Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The Eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on world wide web (WWW '03). ACM, pp 640–651
10. Yamamoto A, Asahara D, Itao T, Tanaka S, Suda T (2004) Distributed pagerank: A distributed reputation model for open peer-to-peer networks. In: Proceedings of the 2004 international symposium on applications and the internet workshops (SAINTW'04). IEEE Computer Society, pp 389–394

11. Jun S, Ahamad M (2005) Incentives in BitTorrent induce free riding. In: Proceedings of the 2005 ACM SIGCOMM workshop on economics of peer-to-peer systems, ser. P2PECON '05. ACM, pp 116–121
12. Sherman A, Nieh J, Stein C (2012) FairTorrent: a deficit-based distributed algorithm to ensure fairness in peer-to-peer systems. IEEE/ACM Trans Netw 20(5):1361–1374. [Online]. Available: doi:10.1109/TNET.2012.2185058
13. Piatek M, Isdal T, Anderson TE, Krishnamurthy A, Venkataramani A (2007) Do incentives build robustness in BitTorrent? In: Proceedings of the 4th symposium on networked systems design and implementation (NSDI 2007). USENIX
14. Feldman M, Lai K, Stoica I, Chuang J (2004) Robust incentive techniques for peer-to-peer networks. In: Proceedings of the 5th ACM conference on electronic commerce (EC'04). ACM, pp 102–111
15. Langville AN, Meyer CD (2004) Deeper inside pagerank. Internet Math 1(3):335–380. [Online]. Available: http://akpeters.metapress.com/content/bn22r01j43g6q8g6/
16. Piatek M, Isdal T, Krishnamurthy A, Anderson TE (2008) One hop reputations for peer to peer file sharing workloads. In: Proceedings of the 5th USENIX symposium on networked systems Design & Implementation (NSDI 2008). USENIX Association, pp 1–14
17. Delaviz R, Andrade N, Pouwelse JA, Epema DHJ (2012) SybilRes: A sybil-resilient flow-based decentralized reputation mechanism. In: Proceedings of the 2012 IEEE 32nd international conference on distributed computing systems (ICDCS '12). IEEE Computer Society, pp 203–213
18. Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: Bringing order to the Web. Stanford InfoLab, Technical Report 1999-66, Nov. 1999, previous number = SIDL-WP-1999-0120. [Online]. Available: http://ilpubs.stanford.edu:8090/422/
19. Varga A (2010) OMNeT++. In: Wehrle K, Günes M, Gross J (eds) Modeling and tools for network simulation. Springer, Berlin, pp 35–59
20. Locher T, Moor P, Schmid S, Wattenhofer R (2006) Free riding in BitTorrent is cheap. In: Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets), pp 85–90
21. De Vogeleer K, Erman D, Popescu A (2008) Simulating BitTorrent. In: Proceedings of the 1st international conference on simulation tools and techniques for communications, networks and systems & workshops, ser. Simutools '08. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, pp 2:1–2:7

Andrei Gurtov received his M.Sc (2000) and Ph.D. (2004) degrees in Computer Science from the University of Helsinki, Finland. He is a Full Professor at Linköping University, Sweden. He is also an adjunct professor at Aalto University, University of Helsinki and University of Oulu. He visited ICSI in Berkeley multiple times. He is an ACM Distinguished Scientist, IEEE ComSoc Distinguished Lecturer and Vice Chair of IEEE Finland section. Andrei co-authored about 200 publications, including 4 books, 5 IETF RFCs, 6 patents, over 50 journal and 100 conference articles. He supervised 12 PhD theses, serves as an editor of IEEE Internet of Things URL: http://gurtov.com.

Joakim Koskela received his Ph.D. in Computer Sciences from Aalto University, Finland in 2015. His research focused on security in distributed systems, which included IETF standardization work and international joint projects on topics such as identity management, privacy, trust- and reputation systems. He is currently working on IoT provisioning systems for the telecommunications industry.

Dmitry Korzun received his B.Sc. (1997) and M.Sc (1999) degrees in Applied Mathematics and Computer Science from the Petrozavodsk State University (PetrSU, Russia). He received a Ph.D. degree in Physics and Mathematics from the St.-Petersburg State University (Russia) in 2002. He is an Associate Professor at Department of Computer Science of PetrSU (since 2003 and ongoing). He was Visiting Research Scientist at the Helsinki Institute for Information Technology HIIT, Aalto University, Finland (2005-2014). In 2014-2016 he performed the duties of Vice-dean for Research at Faculty of Mathematics and Information Technology of PetrSU. Since 2014 he has acted as Leading Research Scientist at PetrSU, originating research and development activity within fundamental and applied research projects on emerging topics in ubiquitous computing, smart spaces, and Internet technology. Dmitry Korzun serves on technical program committees and editorial boards of a number of international conferences and journals. His research interests include modeling and evaluation of distributed systems, mathematical modeling and concept engineering of cyber-physical systems, ubiquitous computing and smart spaces, Internet of Things and its applications, software engineering and programming methods, algorithm design and complexity, linear Diophantine analysis and its applications, theory of formal languages and parsing. His educational activity started in 1997 at the Faculty of Mathematics of PetrSU (now Institute of Mathematics and Information Technology). Since that time he has taught more than 20 study courses on hot topics in Computer Science, Applied Mathematics, Information and Communication Technology, and Software Engineering. He is an author and co-author of more than 150 research and educational publications.