2012

Decision Strategies for a P2P Computing System

Grzegorz Chmaj
University of Nevada, Las Vegas, chmajg@unlv.nevada.edu

Krzysztof Walkowiak
Wrocław University of Technology, krzysztof.walkowiak@pwr.wroc.pl

Journal of Universal Computer Science, vol. 18, no. 5 (2012), 599-622
submitted: 15/12/10, accepted: 28/2/12, appeared: 1/3/12 © J.UCS

Decision Strategies for a P2P Computing System

Grzegorz Chmaj
(Wroclaw University of Technology, Wroclaw, Poland
grzegorz@chmaj.net)

Krzysztof Walkowiak
(Wroclaw University of Technology, Wroclaw, Poland
krzysztof.walkowiak@pwr.wroc.pl)

Abstract: Peer-to-Peer (P2P) computing (also called 'public-resource computing') is an effective approach to perform computation of large tasks. Currently used P2P computing systems (e.g., BOINC) are most often centrally managed, i.e., the final result of computations is created at a central node using partial results – what may be not efficient in the case when numerous participants are willing to download the final result. In this paper, we propose a novel approach to P2P computing systems. We assume that results can be delivered to all peers in a distributed way using three types of network flows: unicast, Peer-to-Peer and anycast. We describe our concept of the system architecture with a special focus on the decision strategies developed for system participants. Using our discrete realtime simulator we evaluate the proposed strategies in various scenarios and present a comprehensive analysis of obtained results. According to obtained results, the Peer-to-Peer flow provides lower operational cost of the computing system compared to unicast and anycast flows. Moreover, an appropriate selection of decision strategy can significantly reduce the operational cost.

Keywords: computing systems, P2P, unicast, anycast, networks, simulation
Categories: C.2.1, C.2.3, C.2.4

1 Introduction

Nowadays, permanently growing need for data processing triggers development of various kinds of distributed computing systems. Peer-to-Peer (P2P) computing systems allow performing distributed computation of tasks, which requires huge processing power that is mostly not available on a single machine. Such tasks are divided into many uniform pieces, which are distributed to separate machines (peers), which perform computation and next send the result back to the central node. The most popular implementation of this idea is the BOINC framework [Anderson, 04], which is used to run projects such as Seti@Home [Anderson, 02], Einstein@Home [Einstein, 10] and many others. The elements of such systems are most often home computers (such as PCs, Apples, or even game consoles such as Sony Playstation). Therefore, these systems are called Peer-to-Peer computing or public-resource computing systems. It must be noted that there is also a different kind of distributed computing systems called Grids [Travostino, 06], [Wilkinson, 09] that consist of dedicated machines with high computing power, e.g., super-computers or clusters, owned by various institutions or corporations. Two mentioned kinds of computing systems (i.e., P2P and Grids) differ in the number of participants – P2P systems

contain many machines (even up to millions [BOINC, 10]), while Grids incorporate most often up to tens of participants [Wyrzykowski, 05], [EGI, 10]. The common feature of Grids and Peer-to-Peer computing systems is the rule, that participants provide their computing power and when they finish the computation – they send the obtained result back to the managing (central) node, where all partial results are combined/analyzed into the final result. The participant usually does not analyze her/his partial result – as she/he would have to examine other partial results to produce the final result. In the case when the system's participant is interested in the final result – she/he simply downloads it from the central node. This schema may cause huge load at the central node and high network traffic leading to network congestion. Therefore, in this paper we focus on a novel P2P computation architecture, where participants both compute source blocks and distribute results to all other participants instead of sending them back to the central node. This way we model computation community, where all members are interested in final results. Note that the architecture was first time introduced in [Chmaj, 10a]. Today's distribution computation systems suffer problems with delivery of the final results to a large number to requesting sites. Authors of Electric Sheep project [Draves, 04] use distributed computing to render artificial forms of life and pinpoint that their central node where partial results are combined into the final result is often overloaded due to many download requests. Our novelty is a distribution computation system, which will be able to perform efficient task processing together with efficient result delivery. To realize data delivery, we propose to use Peer-to-Peer flow and compare its efficiency with unicast and anycast flows. We show that the use of Peer-to-Peer flow can significantly decrease the operational cost of the system. Our approach partially takes inspiration from BitTorrent protocol [Cohen, 03] – we use a kind of a tracker service. The proposed P2P computing system is managed in a distributed manner. Each element of the system (participant or tracker) is to make several decisions to process and distribute data. The performance of the whole system strongly depends on these strategies. Therefore, we propose and examine several decision policies.

To evaluate the proposed computing system and decision strategies we developed a simulator that enables to examine the following issues: simultaneous computing and the result distribution, influence of proposed decision strategies, impact of different kinds of network flows (unicast, anycast, Peer-to-Peer), effect of network parameters such as link speed and computation power. Note that existing simulators (e.g., ns-2, ns-3 and OPNET, [OPNET, 10]) do not provide all required functionality indispensable to analyze all mentioned issues.

Since the data distribution has a significant impact on the performance of the whole system, we compare three kinds of network flows applied for data distribution: unicast, anycast and Peer-to-Peer. The unicast flow is a simple point-to-point transfer and it reflects classical client-server architecture. Data may be fetched only from one node (server). Moreover, the node that downloaded a particular file cannot upload it to any other node. The anycast approach uses special nodes called replicas, which provide data for other peers. The node downloading the file may select any replica to download the file. Anycast flow is a version of the client-server architecture. However, the novelty comparing to pure unicast flow is that the server node (source of data) is deployed in a distributed manner, i.e., the same content is placed in many replica servers spread over the network. Finally, the P2P flow – in contrast to the

client server architecture – assumes that the file can be downloaded from any other node that already possesses the required file. Thus, when using the P2P flow all nodes act both as servers and clients.

The problem researched in this paper may be described as follows. We examine a distributed P2P computing system constructed to provide efficient computation and result dissemination – for more details on the architecture see the next section. The objective to minimize the operational (OPEX) cost of the system including two types of costs, i.e., computation costs and transmission costs. It is worth to notice that tasks computation and results dissemination are mutually influential and should be optimized jointly. Therefore, we propose and examine a set of decision strategies applied in the system as well as we analyze three types of network flows used for data delivery in order to find the best (in terms of the operational cost) configuration. The main motivation behind the research is the fact that currently used solutions do not reflect the computation and dissemination problems at the same time, what leads to inefficient system operation and high costs. Moreover, the centralized approach is the most common approach to manage the system, but as we show in [Chmaj, 10a], it is not an efficient method.

The main contributions of this paper are as follows: (i) detailed discussion about decision strategies developed for the P2P computing system; (ii) experimental investigation considering the influence of decision policies and network flows on system efficiency; (iii) analysis of system's behavior as a function of various network parameters. Note that in our previous paper [Chmaj, 10a], we have presented main concepts of a P2P computing system architecture and we have described the simulation system developed to examine the P2P computing system. We have also briefly presented the decision strategies, which we comprehensively analyze in this paper.

In our previous papers, we have proposed and described ILP (Integer Linear Programming) models, offline heuristic algorithms and discussed their results [Chmaj, 08a], [Chmaj, 08b]. We also analyzed various network flows for the same problem, including the random approaches [Chmaj, 10b].

The remainder of this paper is organized as follows. In Section 2, we describe the proposed P2P computing system. Section 3 includes detailed description of decision strategies developed for the system. In Section 4, we report and discus simulation results. Section 5 presents the related work. Finally, the last section concludes this work.

2 P2P Computing System

The proposed P2P computing system uses the overlay network that provides direct connection between every two elements connected to this network. It is not important how the connections (in our case: between two IP addresses) are established, since the underlying network (e.g., Internet) is responsible for routing. The concept of overlay network is very common in the case of P2P systems [Buford, 09], [Shen, 09], [Steinmetz, 05]. The P2P computing system includes two types of elements:

1. *node* – a regular machine (e.g., PC) that performs the computations and exchange results with other nodes;

2. *tracker* – a central element, which assigns source blocks to nodes. It is also used to provide information related to location of result blocks, what is similar to the BitTorrent tracker concept.

Figure 1 presents the idea of the P2P computing system. There are many nodes and one tracker in the system. Each node is connected to the overlay network using a link having limited upload and download speeds. As nodes are usually private home computers, the network link can be asymmetric DSL connection, local LAN or wireless GSM/3G mobile connection. Thus, we have very wide range of speeds, where often the upload speed is lower than the download speed. Nodes perform both computation and the result distribution. The computation project to be calculated in the system is divided into uniform pieces of the same size called *source blocks*. After processing (of the source block) the obtained output data is called a *result block*. When a node wants to compute a source block, it may obtain it only from the tracker. Thus, a node sends *source block request* to the tracker. If there is at least one available block, the tracker sends the source block to the requesting node. This block is then marked as 'sent to computation' and is not transferred to other nodes.

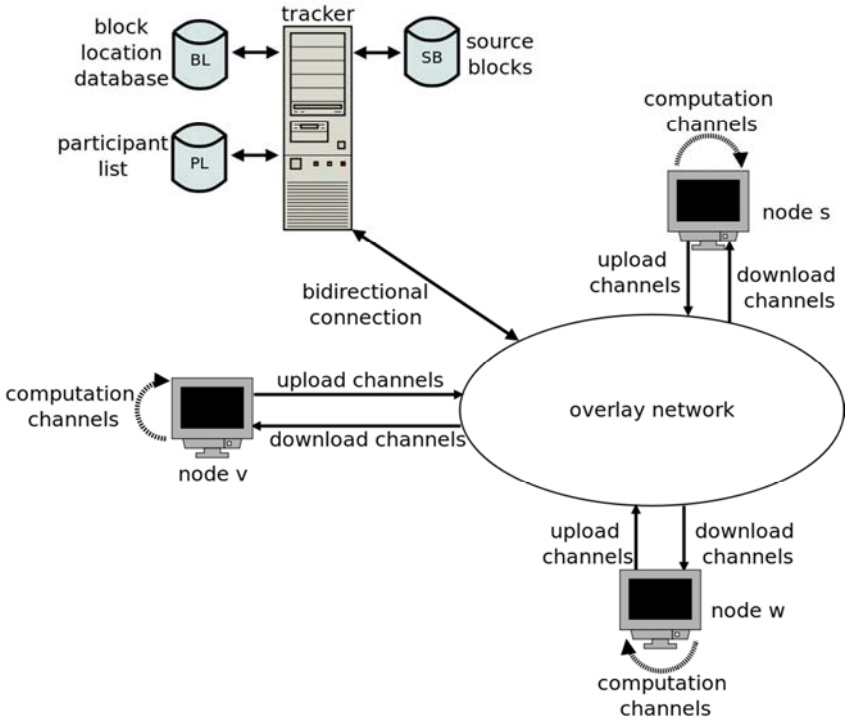

Figure 1: P2P computing system architecture

We assume that system participants are highly interested in all results, so they stay connected when the system works. Failures, block resending and nodes reliability issues are considered as a future work. A node may compute many blocks at the same

time (depending on its CPU power), however we introduce a *fairness* rule, that states: every node has to compute at least one block to become the project participant. Moreover, each participant is interested to obtain all results. Therefore, each block must be transferred to each node (either as a source block or as a result block). The tracker keeps a database containing information about blocks' locations. A node that wants to download the result computed by other node sends a *block location request* to the tracker, in order to obtain list of nodes that store the desired result block. The tracker answers with a *block location list* message including the current location list, which is then processed by the node. To keep the tracker's location list up to date, nodes are to send *tracker update* messages every time the node obtains a new block available to others. This happens in two cases: (1) node has just finished computing the source block, (2) node has just finished downloading a result block from another node. As the tracker knows, which nodes are project's participants (according to the source blocks assignment and messages about computation completion messages) it provides locations only to participants what guarantees the fairness. When a node wants to download a result block, it sends the *download request* message, which can be positively acknowledged by a *download acknowledgement* message. The system presented in this paper uses one tracker – what makes it the central element. However, the tracker service can be deployed in a distributed way, i.e., many trackers may be used and they may be responsible for the same or different data. Our system architecture can be easily modified to consider this case. Thus, the use of one tracker does not make our system *central* - we assume only one tracker due to simplicity reasons.

Let us define remaining terms required to comprehensively describe the proposed system:

- *upload channel* – a node divides its network upload bandwidth into several channels, to be able to manage many simultaneous uploads. Once the channel is reserved to one particular upload, it becomes busy until sending process is finished. The upload channel may be used to send result blocks and various requests.
- *download channel* – like in the case of the upload channel, the download bandwidth is also divided into several channels used later for many concurrent downloads.
- *computation channel* – node processing power is divided into logical channels. We decided to use this way of modeling of the processing power, to keep consistency with the idea of channels used in the context of upload and download bandwidth. One source block may be computed using one computation channel. This assumption we motivate by willingness of having computation performance factor capable to depict number of blocks possible to compute in a certain time. The number of blocks in the system can be large, so the situation when a node is computing a small number of blocks while having free computation channels will be rather rare.
- *message* – nodes and the tracker may interact between each other by sending messages. We distinguish the following kinds of messages in our system:
    - *source block request* is sent from a node to the tracker to inform that the node wants to obtain a new source block for computing;

  - *tracker update* is sent from a node to the tracker to inform that the node possesses a new result block available to be downloaded and this message updates the tracker's location list;
  - *block location request* is sent from a node to the tracker to request the current list of nodes that possess result blocks;
  - *download request* is sent from a node that misses a result block to another node, which possesses the requested result block available for download. It must be confirmed by the *download acknowledgement* message;
  - *download acknowledgement* – this message is a positive reply to the *download request* message;
  - *block location list* is sent from the tracker to a node, requesting the current list of blocks' location, this a reply to the *block location request* message.
- *message queue* – messages are put in a queue from which they are later picked out for further processing. Queues are managed according to global policies described below. Each node has one message queue and one request queue.

Distribution of the results can be performed using three kinds of flows: unicast, anycast and P2P. In the case of unicast flow, the result block can be downloaded only from the node that computed the block. Anycasting assumes that several peers are selected as replica nodes. Each result block is first delivered to all replicas and next other peers can fetch the block from any replica. Finally, in the P2P approach result blocks can be downloaded from any peer that possesses the requested block. Intuitively, we can guess that the P2P concept will provide the best performance in terms of the operating cost, while unicast is expected to yield the highest cost. This hypothesis is confirmed by simulations in Section 4. However, we should also enumerate potential drawbacks of proposed data distribution methods. First of all, anycast and especially P2P flows generate additional signaling traffic in the network. Nevertheless, since the result blocks are mostly much larger than signaling messages, the impact of signaling traffic can be neglected. Moreover, anycasting and P2P approaches require more complicated client software, however taking into account high computational power of current computers again this drawback is ignored. Finally, using anycast and P2P flows, peers are more involved in uploading, thus the upload capacity of access links is utilized. But, this aspect can be used to tackle the problem of fairness in the system. For more information on the system architecture of our system refer to [Chmaj, 10a].

3 Decision Strategies

The proposed P2P computing uses a distributed management. Each participant makes decisions individually according to her/his own knowledge and local constraints. We identify the following three decisions of the system:

- *missing block selection* (decision A) – a node indicates one block to be downloaded and this is selected among all missing blocks.

- *source node selection* (decision B) – a node specifies another node that possesses the selected missing block in order to send the download request.
- *request selection* (decision C) – each node receives download requests from other nodes and queues these requests for later processing. The decision determines how such a queue is processed.

In the case of decision A we propose the following strategies (Fig. 2):

- *First-Missing* policy assumes that the first missing block is attempted to download. This works as in many P2P systems, where blocks are requested in order they combine into desired file.
- *Rarest-Missing* policy – a node finds out the rarest result block of all its missing blocks. This block is requested to download. The accuracy of the selection is limited to knowledge available at the node at the time when the decision is made. This concept is widely discussed in the scope of various Peer-to-Peer systems [Buford, 09], [Shen, 09], [Steinmetz, 05].

First-Missing:
Let Missing(v) return a set of blocks that are missing at node v. Let MinId(A) return an index of a block with a smallest value of the identification number included in set A.

```
int First-Missing(int v)
 {
  return MinId(Missing(v));
 }
```

Rarest-Missing:
Let function RarestBlock(A) return the rarest block among blocks included in set A. This information is provided by the tracker, which has the global knowledge on the blocks' availability on each node.

```
int Rarest-Missing(int v)
 {
  return RarestBlock(Missing(v));
 }
```

Figure 2: Missing block selection decision strategies in a pseudocode

In the decision B we apply the following strategies (Fig. 3):

- *First-on-the-List* policy assumes that the first node on the list of desired block owners is selected. List of nodes having desired block is obtained from the tracker and it is not ordered.
- *Cheapest-Owner* policy allows the node to analyze all nodes possessing a particular block and select the node according to a selected criterion, e.g., cost, link speed, number of hops, etc. This way we obtain a system, which works with regard to optimizing one of many criterion factors.

Finally, in the case of the decision C we introduce the following policies:

- *First-Available* is the implementation of the FIFO queue, where the first received request is processed first.
- *Cheapest-Available* policy assumes that the node having sufficient resources (e.g., bandwidth) may select the request freely from all available in queue. Attractiveness is defined in the same way as for *Cheapest-Owner* policy. If we assume, that our criterion will be the amount of data sent by requesting node, then this policy will resemble the tit-for-tat algorithm used in BitTorrent [Cohen, 03], [Buford, 09], [Shen, 09].

First-on-the-list:
Let function GetOwners(b) return the set of nodes which own block b (thus may send this block to other node). This information is provided by tracker according to its knowledge. This set is not sorted and nodes are placed in order update messages arrive to tracker. Let FirstFromSet(A) return the first element from set A, and b denotes block which is to be download by node v.

```
int First-on-the-list(int b)
 {
 return FirstFromSet(GetOwners(b));
 }
```

Cheapest-owner:
Let function MinCost(v, A) return the id of node, from which transfer cost is smallest to node v. Returned node id is selected among node set A.

```
int Cheapest-Owner(int v, int b)
 {
 return MinCost(v, GetOwners(b))
 }
```

Figure 3: Source node selection decision strategies in a pseudocode

First-available:
Let FirstRequest(v) return an index of a request which was put into the queue at the earliest time among all requests in the queue. Queue is owned by node v.

```
int First-Available(int v)
 {
 return FirstRequest(v)
 }
```

Cheapest-available:
Let MinRequest(v) return the id of a request, which was sent by a node having smallest transfer cost to node v. Request is chosen among requests present in the queue owned by node v.

```
int Cheapest-Available(int v)
 {
 return MinRequest(v);
 }
```

Figure 4: Request selection decision strategies in a pseudocode

Each participant of the system is configured to use the same set of decisions strategies. Since there are 2 strategies for each of 3 decisions, it makes 8 possible combinations of the system configuration. The simulation comparison of these scenarios is presented in the next section.

4 Results

To examine the decision strategies proposed above, we developed our own simulation system written using C++, STL and compiled using gcc (for Linux environment) and Visual Studio 2003 (for MS Windows environment). We did not apply any other simulator (e.g., ns-2, ns-3 or OPNET), since none of existing systems provides the required functionality that enables simulating of both computation and result distribution in overlay network with the use of various network flows (unicast, anycast and P2P) and a range of decision strategies. For a detailed description of the simulation system see [Chmaj, 10a].

As the main performance metric, we propose to use the operational (OPEX) cost of the computing system including two elements: computing cost and result distribution costs. Note that in the case of a P2P computing system created of individual machines, the OPEX cost is much more significant than the CAPEX. To observe the influence of proposed policies on the system performance, we run experiments using the simulation system. We use networks having various size in

terms of number of nodes number of blocks and number of iterations. Using the iteration parameter, we model the network operating time, in which all computations and transfers must be completed (we model our problem as time-constrained). Other parameters of the networks (access link capacity, processing power, transfer cost and processing cost) are selected randomly according to parameters of real overlay systems, , e.g. for the case of download speed, simulation values have been set to 10-160 units, where 1 unit = 100kb/s, what gives us simulated network having download links from the range of 1MB/s to 16MB/s. This random value assignment reflects how real network is built – different machines have different parameters, but enclosed in some range. For example, the processing power, which may be different between machines influences the time of block computation in our system. Source block is the piece of data to be computed and we assume that some certain amount of processing work has to be done to compute the source block. Thus, in our simulations a machine with a higher computation power will compute the source block faster than a machine with worse processing ability. To model the cost of each operation, we use the values respective to real network parameters. Processing cost reflects the energy and hardware expenses, while the network costs we model as the price of transfer of one source block between two machines using the overlay network. Our model is universal and transfer cost may reflect many factors, such as: distance between two machines, provider's network access cost and others. For the researched networks, we take cost values from a specified range to model various sizes of the network ranging from small systems developed within one small region to networks covering intercontinental area.

The first goal of simulations is the comparison of decision strategies described in Section 3. Since there are 2 strategies for each of 3 decisions, it makes 8 possible combinations (Table 1). For each network and each combination of policies we investigate three types of network flows: unicast, P2P and anycast.

Combination/ experiment identifier	**Strategies used**		
	Decision A *missing block selection*	**Decision B** *source node selection*	**Decision C** *request selection*
P0	First-Missing	First-on-the-List	First-Available
P1	Rarest-Missing	First-on-the-List	First-Available
P2	First-Missing	Cheapest-Owner	First-Available
P3	Rarest-Missing	Cheapest-Owner	First-Available
P4	First-Missing	First-on-the-List	Cheapest-Available
P5	Rarest-Missing	First-on-the-List	Cheapest-Available
P6	First-Missing	Cheapest-Owner	Cheapest-Available
P7	Rarest-Missing	Cheapest-Owner	Cheapest-Available

Table 1: Combinations of decision strategies

First, we report detailed results obtained for three selected networks, as general trends are very similar for other examined networks. The tested networks have the following sizes: 100 nodes and 200 blocks; 127 nodes and 254 blocks; 157 nodes and 314 blocks. The number of iterations is set to 15. For each individual case in terms of the network, decision strategy and flow type, we repeated the experiment 10 times to

enable statistical analysis of the results. In Figs. 5-7, we present the average values of the OPEX cost as a function of decision strategy and network flow type for each network. Tables 2-4 include corresponding results showing lengths of 95% confidence intervals for each particular series of experiments. Note that a particular case (network, decision strategy and flow type) with no feasible solution is reported in the corresponding table cell as N/A.

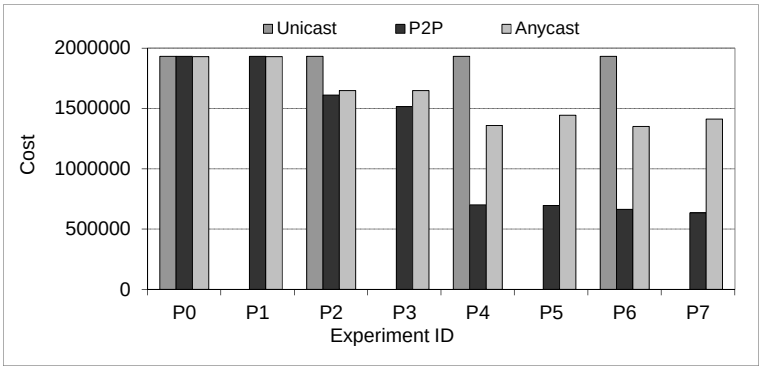

Figure 5: Operating cost as a function of decision strategies and network flows for network with 100 nodes

Flow	**Experiment ID (Combination of decision strategies)**							
	P0	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
unicast	0	N/A	0	N/A	0	N/A	0	N/A
P2P	3851	3585	1644	5357	694	973	583	445
anycast	2159	2513	2018	1530	1295	2163	1111	2371

Table 2: Length of 95% confidence intervals for the OPEX cost (network with 100 nodes)

Comparing pairs of experiments in Figs. 5-7, i.e., P0 vs. P1, P2 vs. P3, P4 vs. P5, P6 vs. P7 – we can see that the use of the *Rarest-Missing* policy instead of *First-Missing* policy does not introduce significant difference for P2P and anycast flows. This observation holds for all examined networks. A combination of policies *First-on-the-List* and *First-Available* (in practice: no optimization) causes (for some networks) slightly higher cost for the P2P flow comparing to unicast and anycast flows. The use of the *Rarest-Missing* policy for the unicast flow most often leads to the lack of feasible results.

Deep investigation of simulation results showed us that the *Rarest-Missing* policy creates the starvation effect [Mathieu, 06] – some nodes cannot gain all result blocks (see experiments P1, P3, P5 and P7 in Figs. 5-7). This happens because nodes obtain the same information about block location and thus, they request the same set of blocks. Therefore, they compete for the same small set of "rarest" blocks, what is most influential in unicast flow, where a block may be fetched only from a node, which computed it. In effect, starved nodes send many download requests at early

stage of simulation, but they do not receive requested blocks because these blocks are "popular" at this time and such blocks may be fetched from only one node. As download requests are without any reply – starved nodes do not make any transfers at early stage and even if they get "rare" blocks later – the queue is long and they cannot get all blocks. P2P flow is fully resistant for starvation effect caused by *Rarest-Missing* policy – it does not occurred for all researched networks, and it rarely happened for anycast flow. Starvation effect was widely discussed in [Mathieu, 06].

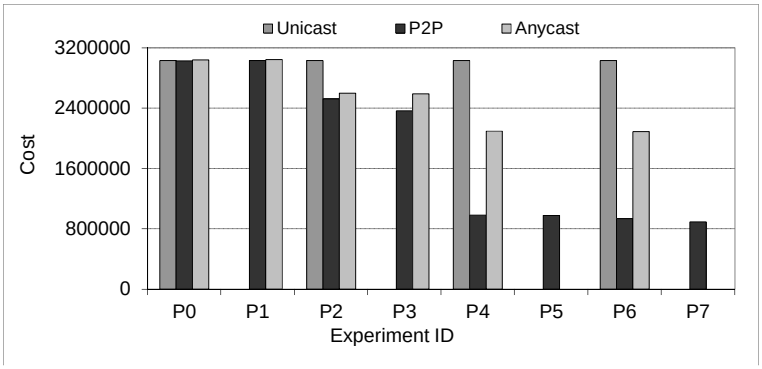

Figure 6: Operating cost as a function of decision strategies and network flows for network with 127 nodes

Flow	**Experiment ID (Combination of decision strategies)**							
	P0	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
unicast	0	N/A	0	N/A	0	N/A	0	N/A
P2P	5094	5055	5389	3079	985	742	798	272
anycast	2303	2142	3398	2611	1455	N/A	965	N/A

Table 3: Length of 95% confidence intervals for the OPEX cost (network with 127 nodes)

Comparing Tables 2-4 against corresponding results presented in Figs. 5-7, we can notice that lengths of 95% confidence intervals are very short, i.e., our simulator provides stable results. Notice that in the case of unicast flows, the length of 95% confidence interval is always 0. This follows from the construct of the simulation systems. In more details, when the system starts each computing node is allocated with one source block to fulfill the fairness assumption of our system. Next, when a node processed its block (what is a function of the node processing power), it requests the next block. Therefore, the block allocation process is deterministic and generates always the same cost. In the case of unicast flows the result block distribution also generates always the same cost, since each requesting peer can download the result block only from the node that computed that block.

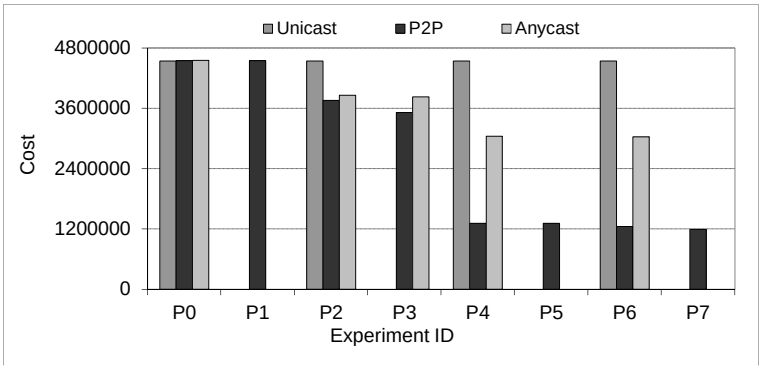

Figure 7: Operating cost as a function of decision strategies and network flows for network with 157 nodes

Flow	**Experiment ID (Combination of decision strategies)**							
	P0	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
unicast	0	N/A	0	N/A	0	N/A	0	N/A
P2P	5041	5729	4266	4572	1477	887	1071	559
anycast	2762	N/A	3196	3176	3434	N/A	2077	N/A

Table 4: Length of 95% confidence intervals for the OPEX cost (network with 157 nodes)

To make a more general comparison of strategies for a larger set of networks, we generated a set of 20 networks with the following parameters: number of nodes V is in range 50-107, number of blocks B in range 75-161. The number of iterations is set to 15. We define a *relative performance* metric of an experiment (combination of decision strategies) as the ratio between the cost obtained for this particular experiment and the minimum cost yielded for the best set of decision strategies. For instance, if for a particular network and type of the network flow the experiment P0 gives cost of 500 and the lowest cost value 400 is obtained for experiment P7, then the relative performance metric of P0 is 20% = (500 – 400) /500, while in the case of combination P7 the metric is 0%.

However, not all experiments result in a feasible output, i.e., in some cases the selected decision strategies and the type of network flow do not allow performing the computation and data distribution in a given number of iterations. For instance, the use of *Rarest-Missing* policy (combinations P1, P3, P5 and P7) sometimes makes unicast and anycast flows unable to return a complete solution. This happens because of starvation effect explained above. Additional experiments showed that even significant increase of iterations parameters (i.e., the network operation time) did not fix this problem – so additional mechanisms are required in this case. Detailed results are shown in Table 5. For each type of the network flow we bold the best combination of decision polices. Note that for the P2P flow, in all cases a feasible result is obtained.

Flow	Experiment ID (Combination of decision strategies)							
	P0	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
unicast	**20**	3	**20**	1	**20**	3	**20**	4
P2P	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
anycast	**20**	**20**	**20**	13	**20**	19	**20**	1

Table 5: Number of feasible results for various combinations of decision strategies and network flows

In Table 6, we report the average results of the relative performance metric for each type of network flows. The best combination is bolded. To show statistical analysis of the results, Table 7 includes respective lengths of 95% confidence intervals. If the number of analyzed results (networks) is lower than 10 (see Table 3) we do not provide the length of 95% confidence interval.

Flow	Experiment ID (Combination of decision strategies)							
	P0	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
Unicast	0.84%	1.86%	0.84%	1.49%	0.95%	0.71%	**0.64%**	2.01%
P2P	38.37%	38.30%	4.86%	4.31%	30.19%	28.55%	1.85%	**0.08%**
Anycast	16.92%	16.78%	1.90%	2.94%	8.68%	9.82%	**0.05%**	2.67%

Table 6: Average values of relative performance metric as a function of network flows

Flow	Experiment ID (Combination of decision strategies)							
	P0	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
Unicast	0.40%		0.36%		0.40%		0.33%	
P2P	2.11%	2.18%	0.67%	0.77%	2.09%	1.93%	0.59%	0.15%
anycast	1.17%	1.33%	0.59%	0.92%	1.05%	0.97%	0.08%	

Table 7: Length of 95% confidence intervals for the relative performance metric

Table 8 shows the average score of each experiment (combination of decision strategies) calculated in the following way. The combination that yields the lowest cost of a particular network gets the score 7, the second combination receives score 6, etc. If there is no feasible result, the score is 0.

In the case of the unicast flow, the change of strategies does not have significant influence on the final result. The average value of the relative performance metric is always below 2.01%. For the P2P flow, selection of strategies is very influential – the differences between results reached even 44% for one of the networks. The best performance is achieved for experiment P7, where results are minimal in the case of 19 networks comparing to other experiments, and average value is 0.08%. Thus, policies *Cheapest-Owner* and *Cheapest-Available* are the best for the P2P flow. The anycast flow is less sensitive to selected strategies – the largest value of the relative

performance metric is 21%. Analysis of results shows that in the case of anycast flows, the *Rarest-Missing* policy causes a lack of solution for some networks. The best result is achieved for experiments P6 and P2.

Flow	Experiment ID (Combination of decision strategies)							
	P0	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
Unicast	5.4	0.7	5.3	0.2	4.9	0.8	**5.7**	0.8
P2P	0.5	0.5	4.4	4.7	2.0	3.0	6.0	**6.9**
Anycast	1.9	1.9	5.9	3.5	4.1	3.3	**6.8**	0.2

Table 8: Average values of the experiment score as a function of network flows

The next goal of experiments is to examine the system's behavior as a function of various network parameters including link capacity, processing power, number of nodes, and number of blocks. We present results obtained using the best (cheapest) configuration of the system in terms of decision strategies and flow type, i.e., combination P7 (*Rarest-Missing*, *Cheapest-Owner* and *Cheapest-Available*) and the P2P flow. We present detailed results obtained for four exemplary (selected) networks (50 nodes and 75 blocks; 68 nodes and 102 blocks; 89 nodes and 134 blocks, 107 nodes and 161 blocks). The number of iterations is set to 15. However, results yielded for other networks have shown similar trends.

First, we focus on the upload capacity limit. The methodology of the experiment is as follows. We manipulate the upload capacity to find the minimal feasible values of the limits that enable to run the system (process all source blocks and distribute all results blocks). Next, starting from these feasible values, we increase the upload capacity of each node by 1 unit and repeat this procedure until a significant change is observed. Note that – as above – for each individual case we repeat the same unique experiment 10 times to enable statistical analysis of the results. In Fig. 8, we report average values of the cost as a function of the upload capacity increase for four tested networks.

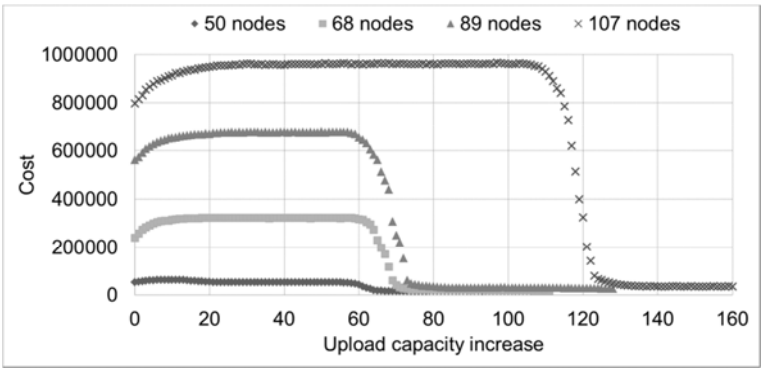

Figure 8: Operating cost as a function of upload capacity increase

Notice that all presented curves follow the same shape. Initially, we observe a slight growth of the cost. Next, the cost remains stable until a sharp decrease occurs. Finally, the cost converges to stable value. A little surprising is the initial slight growth of the cost. This can be explain as follows. At the beginning (i.e., small values of the upload capacity), only a relatively minor part of the solution space is feasible and the obtained OPEX cost reflects these limits. When the upload capacity grows, more possible transfers are available. Since the applied strategies use some stochastic elements, the OPEX cost slightly grows. Next for some time the cost remains stable. The significant drop of the cost and its stabilization at the low value occurs when the system have enough upload capacity that only the cheapest nodes participate in sending blocks and the optimization process is simple.

Fig. 9 shows results with 95% confidence intervals obtained for a 68-node network. For other tested networks, the statistical analysis yields similar trend, i.e., the results are relatively stable in each repetition of the same experiment.

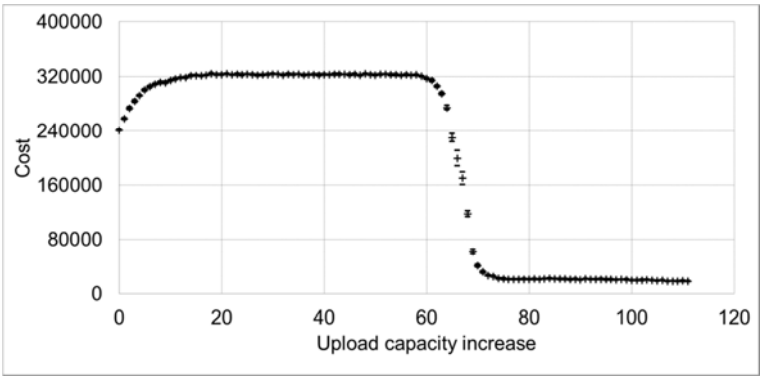

Figure 9: Operating cost with lengths of 95% confidence intervals as a function of upload capacity for a network with 68 nodes

The second analyzed parameter is the download capacity. The procedure is analogous to the upload capacity parameter, i.e., first we find minimal feasible values and next increase the capacity by 1 unit in each node. Obtained results show that the system performance (OPEX cost) generally does not depend on this parameter, i.e., increasing of this parameter starting from the feasible value does not change the OPEX cost.

Similar observation as in the case of download capacity can be noticed for processing power limits. Again, increasing this parameter from the minimal feasible values does not influence the OPEX cost. This can be explained by the fact that above certain limits level, additional computing resources will not be consumed anyway. The system is designed to perform computations as soon as this is possible – to make computation results available early for upload and dissemination. Therefore, the uniform reduction of computing limits is changing the system from the one having more resources than it needs, almost directly to the one suffering lack of computing resources (infeasibility). Thus, we do not observe the OPEX change for computation limits manipulation.

The next examined parameter is the number of nodes. The procedure of the experiments is as follows. Starting from the original network, we increment the number of nodes by 2 until the system becomes infeasible, i.e., when the number of nodes exceeds the number of blocks, according to the fairness assumption guaranteeing that each node must compute at least one block to become the system participant. For each case, we repeat the same experiment 10 times. Fig. 10 shows average values of the cost obtained for all tested network, while Fig. 11 presents statistical analysis (lengths of 95% confidence intervals) for a network with 107 nodes. We can observe that the OPEX cost slightly grows with the increase of the number of nodes. This mainly follows from two assumptions of the considered P2P system. First, each node is to receive all result blocks, therefore all blocks must be delivered to new added nodes, what causes the distribution cost growth. Second, due to the fairness constraint, each new node must be allocated with at least one block to be processed what in many cases increases the processing cost.

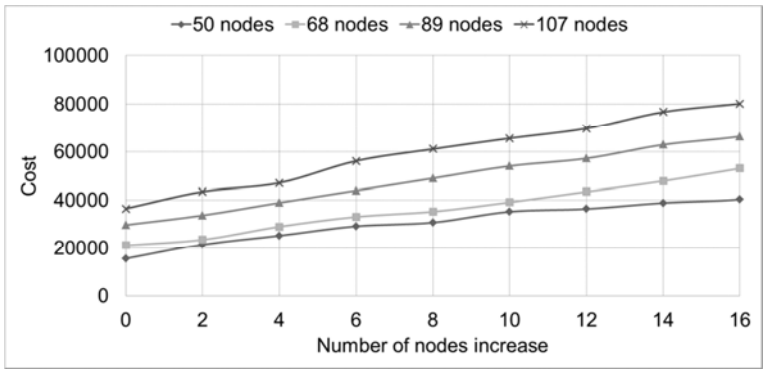

Figure 10: Operating cost as a function of nodes number

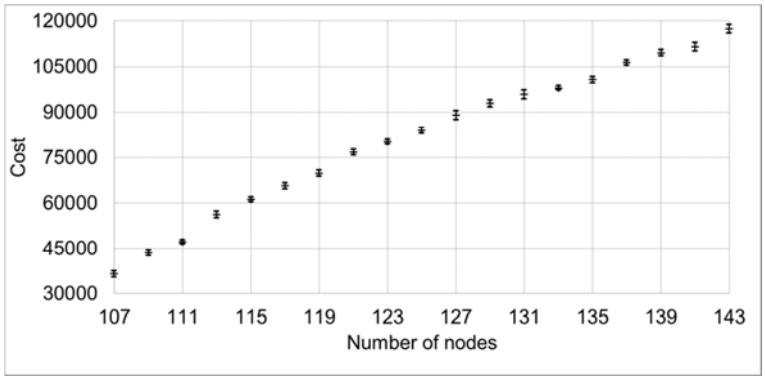

Figure 11: Operating cost with lengths of 95% confidence intervals as a function of nodes number for a network with 107 nodes

The last analyzed parameter is the number of blocks. The methodology of the experiment is analogous to the previous case (number of nodes), i.e., starting from original networks we generate a new network incrementing the number of blocks by 5. In Fig. 12 the average results obtained for all tested network are reported. Fig. 13 includes detailed results with statistical analysis yielded for a network with 89 nodes.

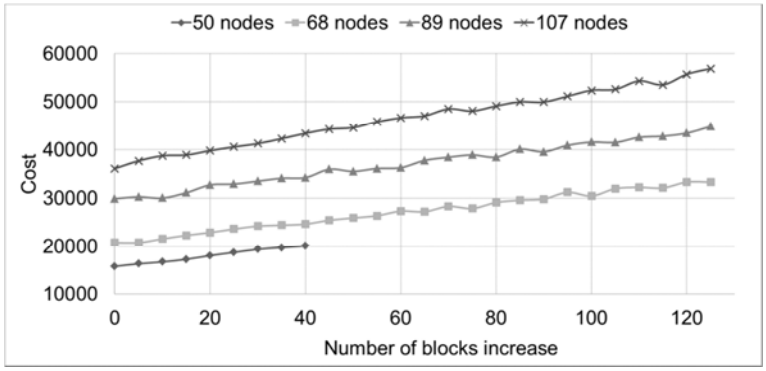

Figure 12: Operating cost as a function of blocks number

Notice that increasing the number of blocks (i.e., problem size) causes the growth of the OPEX cost. This intuitive observation can be explained by the fact that both kinds of costs (processing and transmission) grow with the increase of blocks, since new blocks must be processed and next delivered to all nodes. When the number of blocks exceeds a particular value, the network becomes infeasible, as there is not enough network capacity to serve increasing traffic and computing needs. Infeasibility occurs at various points according to parameters of examined networks.

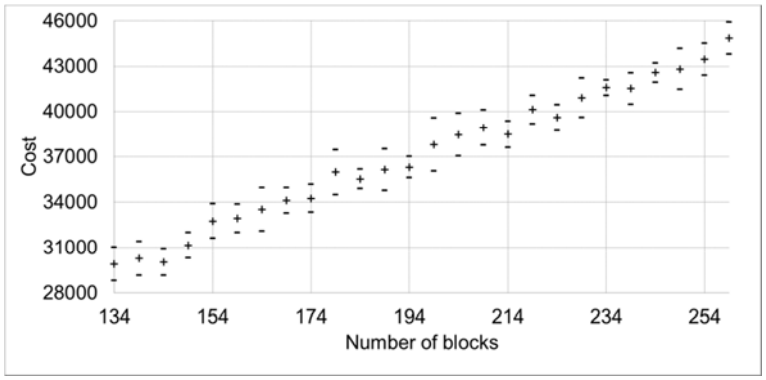

Figure 13: Operating cost with lengths of 95% confidence intervals as a function of blocks number for a network with 89 nodes

Our research shows that the key issue for the OPEX cost optimization is to select the appropriate decision strategy and the network flow type. The most influential

network parameter is the upload capacity, while download capacity and processing limit do not have much impact on the cost. Also network architecture is influential, i.e., nodes with very bad network parameters (high network costs, weak network connections) make the OPEX high, as they cannot be excluded and must take part in the processing. Overall optimization is very complex and must include many parameters, which influence on OPEX is not straight.

5 Related Work

The authors of [Foster, 03] address issues related to Grids and P2P systems. Various aspects including common features and differences between both kinds of systems are discussed. Moreover, the authors propose a system based on these two architectures. Another approach to merge Grid and P2P concepts is described in [Subramanian, 05]. The authors propose a system called IBM Download Grid, which aims to optimize data delivery with the use of a Grid architecture. Also authors of [Uppluri, 05] propose to implement P2P as the transport layer for Grid systems. They propose their own implementation of Gnutella protocol without taking the computation part into the consideration.

P2P systems are considered as an efficient way to transfer data in networks what is investigated in many papers. In [Wu, 05] various approaches to P2P algorithms are discussed. The authors propose CSFD (Centrally Scheduled File Distribution) protocol aiming to minimize total delivery time of a file, which is initially available at only one node (a file is to be delivered to all nodes of the P2P system). CSFD is also compared with the BitTorrent protocol. A similar approach is described in [Ganesan, 05], where the authors propose an optimization model and solution algorithms including a random approach. The study of P2P applications' efficiency is addressed in [Yang, 04]. The authors describe a deterministic model and provide an analytical discussion related to network delay, branching model and other aspects.

Distributed computing is frequently considered in the context of task scheduling, since task scheduling is a vital issue for efficiency of distributed systems. Authors of [Nabrzyski, 04] take into consideration many scheduling criteria, however they pinpoint that their model uses many simplifications. Scheduling is widely described in the literature, e.g., economic algorithms are addressed in [Fujimoto, 04], relations between scheduling techniques and RMS (Resource Management Systems) are presented in [Buyya, 02], task allocation as the form of multidimensional knapsack algorithm is discussed in [Vanderster, 09]. Triana is the distributed computing problem solving system, used (among many things) for visualization of galaxies formation. It is based on decentralized (unlike centralized BOINC) network with Peer-to-Peer flow and inter-node direct communication (without using centralized messaging broker). Triana peer receives the task in the form of the script, and after computation sends the result to Triana client or to another node. This brings the opportunity to compute task using distributed system, but still lacks the possibility to deliver results to all system participants (or the large group of them). The advanced resource aggregation techniques are presented in [Moca, 10]. Authors described decision models for this task applied to distributed systems and propose two-mechanism functionality: resource discovery and partner selection. Thanks to it, resources may be found in the structure and be aggregated at desired node. The

efficiency criterion is defined as resource aggregation efficiency across the network where nodes delivery some resources. Note that in our paper we consider one metric for task scheduling – the choice of a node, to which the computation task will be allocated. The novelty of our approach is to optimize the overall system at all stages of its work: from tasks allocation to result delivery, keeping the regard to OPEX cost at each stage.

Currently, computer networks mostly use four types of network flows: unicast, Peer-to-Peer, anycast and multicast. The unicast approach is widely used in such protocols as IP, MPLS [Ahuja, 93], [Pióro, 04]. At early times of Internet unicast was the only network flow used in protocols like TCP, HTTP, SMTP, FTP and it still remains very popular, although sharing multimedia and popularity of streaming media force to look for another transmission approaches, such as anycast, multicast and Peer-to-Peer. Streaming strategies in Peer-to-Peer networks were described in [Liu, 08]. Authors describe the following approaches: "client-server", "random P2P streaming", "smallest number of hops in P2P" and "maximum profit in P2P" together with research methodology and results. Issues of Peer-to-Peer streaming based on IPTV are presented in [Hei, 08], analysis of Joost Internet TV is described in [Moreira, 08]. The authors of [Conklin, 01] compare various methods of media streaming. Unicast flow is popular subject of network traffic optimization. The most of unicast problems presented in literature are classified as Linear Programming or Mixed Integer Programming [Pióro, 04], for which optimization techniques are proposed as solution (simplex, branch and bound, etc.). This approach allows getting optimal or feasible solution.

P2P systems are not only devoted to file sharing – they can link resources of many kinds, like: processing power, sensors, inter-user communication, etc. However, exchange of files is the most popular application of P2P systems and there are many protocols and systems designed to meet this usage. The most popular of them are: Napster, Gnutella, Kazaa, eDonkey and BitTorrent [Buford, 09], [Shen, 09], [Steinmetz, 05], [Tarkoma, 10]. Another application of the P2P concept is distributed database systems – their main idea is to spread database over many machines [Tjoa, 05]. Peer-to-Peer systems often implement strict rules of fairness – to avoid domination of the whole system by a single peer. Fasttrack protocol characterizes each node by a fairness ratio, which is computed as a relation of data sent by a node to amount of data downloaded by the node. Nodes, which do many downloads without uploading are treated as parasites – they are placed at the end of queue of nodes willing to download particular data piece. A popular model of fairness implementation is tit-for-tat strategy [DeFigueiredo, 07]. This approach assumes that the node may download data only while it is simultaneously uploading data. There are two kinds of tit-for-tat strategy: *direct reciprocity* – node v may download from node w only when v is sending to w; *indirect reciprocity* – node v may download from node w only when any node is sending to w. Strategies of cooperation between nodes in Peer-to-Peer network are described in [Schlosser, 06]. *Random Chunk* strategy (also used in eDonkey) is based on assumption that node does not request pieces of a desired file in any order – random choice is used. Moreover, nodes serving file pieces are queuing download requests using FCFS (First Come, First Served) schema. The same way of queuing is used in the *Least-Shared-First* strategy, but in this case node selects missing file piece according to its popularity in network, i.e., least popular

pieces are requested first. This approach minimizes the problem of unpopular pieces, which are hard to download. Authors of [Schlosser, 06] conclude, that the *Random Chunk* strategy performs well unless nodes with complete file do not disconnect shortly after completing download whole file – then least shared pieces become very hard to get. As the solution for this case authors propose the *Least-Shared-First* strategy, which works well with parasite scenarios, but introduces "last block problem" – solved by CygPriM strategy also suitable for parasite scenarios.

The most popular P2P file sharing protocol is BitTorrent – it is widely used among private users as well in science, where it is considered as efficient way to solve data delivery problems [Cohen, 03], [Buford, 09], [Shen, 09], [Steinmetz, 05], [Tarkoma, 10]. It assumes a special *tracker* role assigned to one node, which acts then as a simple manager and knowledge base. This role may be assigned to any node from the given system. Fairness in BitTorrent protocol is implemented using the tit-for-tat policy. Files are divided into blocks having uniform size and act as basic data units in system – listed in special metadata .torrent files containing also control information. Tracker may be considered as central system node, but it is worth to notice that this role covers only given metadata file (of course one tracker may serve its role for many metadata files). This way one P2P system may contain many trackers managing different (or even the same) files, what makes BitTorrent a non-central architecture. Although popular BitTorrent trackers grouping many users – such as Demonoid or PublicBt face the problem of centrality – they are easy to be turned off by digital right management companies. This problem pushed owners of ThePirateBay to work on structured approach of tracker, which spreads knowledge base (concentrated at tracker in original approach) among many nodes. Note that in our concept of the computing system, we follow BitTorrent ideas, e.g., tracker idea, data in form of blocks, periodic update of knowledge base and tit-for-tat policy. These ideas more or less similar to pure BitTorrent implementation are also used in many P2P and distributed computation systems.

A network applying the anycast flow contains nodes having special role – called *replicas*. Each replica provides the same content. A requesting node chooses one of the replica according to some criterion, e.g. network link speed, geographical distance, etc. Successful implementation of anycast flow is a sophisticated task and requires solving such problems as: replica localization, replica evaluation, data consistency, request routing, accounting and routing [Rabinovich, 98]. Authors of [Zhang, 04] propose ARMM protocol (Anycast Routing protocol based on Multi-Metrics). In contrary of traditional approach, where metric is established in client-to-replica direction, ARMM uses reverse direction: replica-to-client. This way the route is set as best path from replica to downloading node regarding transfer speed. Described research showed, that ARMM provides high efficiency, especially in networks transferring big amounts of data. An example of anycast based system is CDN (Content Delivery Network), which delivers data on behalf of original servers, e.g., Akamai. Original files are placed on replicas located across the network. For each download request, CDN tries to find closest location having desired data [Hofmann, 05]. The anycasting in IP networks is described in [Metz, 02] together with practical implementations – such as DNS address resolving and IPv4 to IPv6 gate. Authors of [Sarat, 06] examine benefits following from the anycasting comparing to a traditional DNS implementation. Various configurations of anycasting

are reviewed and investigation shows that the use of anycast decreases time of DNS response and improves accessibility to nameservers. The author of [Walkowiak, 06] addresses problems related to static optimization of anycast flows and proposes an effective heuristic algorithm.

Our problem could be also applied to sensor networks, in the case when sensors would require receiving results from other sensors. The most common form of the sensor network gathers the data and delivers it to a processing node (also including the data processing at the sensors). This resembles our problem, but we consider the optimization problem as more general, including the computing and disseminating processes at the same time, while sensor networks focus on data processing and delivering to managing node, without the optimization of inter-node processing/communication.

6 Conclusions

In this paper, we described our approach to P2P computing systems. The main novelty of our idea is that the system performs computation and the result distribution at the same time. We showed how to avoid network congestion that can be caused by distribution of results to system participants. Similarly as in pure P2P systems, the computing system uses various decisions, for which we proposed and investigated several strategies. The described system was implemented in a realtime discrete simulation system. Research experiments proved that suitable choice of a right strategy is fundamental for the system's efficiency. The difference between investigated policies (expressed as operating cost) reached even 40%. In experiments we applied three network flows: unicast, Peer-to-Peer and anycast. The P2P flow together with suitable strategy selection yielded the best efficiency of the computing system (up to 60% better than the worst case). Moreover, some policies applied to unicast and anycast flows caused the starvation effect, leading to lack of final result. Thus, as future work, we propose to implement anti-starvation mechanisms for unicast and anycast flows. Moreover, other future directions are as follows: extend simulation system to handle dynamic change of replica status during simulations, dynamic join/leave of nodes and more sophisticated policies comparing to policies presented in this paper.

Acknowledgements

This work was supported by National Science Centre (NCN), Poland, under the grant, which is being realized in years 2010-2013.

References

[Ahuja, 93] Ahuja, R., Magnanti, J., Orlin, J.: Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Englewoof Cliffs NJ, 1993

[Anderson, 02] Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@home: An Experiment in Public-Resource Computing, Communications of the ACM, Vol. 45, No. 11, 2002

[Anderson, 04] Anderson, D.: BOINC: A System for Public-Resource Computing and Storage, Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, 2004

[BOINC, 10] Statistics for projects based on BOINC: http://www.boincstats.com/

[Buford, 09] Buford, J., Yu, H., Lua, E.: P2P Networking and Applications, Morgan Kaufmann, 2009.

[Buyya, 02] Buyya, R.: Economic-based Distributed Resource Management and Scheduling for Grid Computing, School of Computer Science and Software Engineering, Monasch University, Melbourne, 2002

[Chmaj, 08a] Chmaj, G., Walkowiak, K.: Heuristic Algorithm for Optimization of P2P-based Public-Resource Computing Systems, Lecture Notes in Computer Science, Vol. 5375, Springer Verlag, 180-187, 2008

[Chmaj, 08b] Chmaj, G., Walkowiak, K.: Data Distribution in Public-Resource Computing: Modeling and Optimization, Polish Journal of Environmental Studies, Vol. 17, No. 2B, 11-20, 2008

[Chmaj, 10a] Chmaj, G., Walkowiak, K.: A P2P computing system for overlay networks, Future Generation Computer Systems, doi: 10.1016/j.future.2010.11.009, 2011

[Chmaj, 10b] Chmaj, G., Walkowiak, K.: Random Approach to Optimization of Overlay Public-Resource Computing Systems, International Journal of Electronics and Telecommunications, No. 1, Vol. 56, 53-59, 2010

[Cohen, 03] Cohen, B.: Incentives Build Robustness in BitTorrent. Proceedings of the Workshop on Economics of Peer-to-Peer Systems, 2003

[Conklin, 01] Conklin, G. J., Greenbaum, G. S., Lillevold, K. O., Lippman, A. F., Reznik, Y. A.: Video coding for streaming media delivery on the Internet, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, 2001

[DeFigueiredo, 07] DeFigueiredo, D., Venkatachalam, B., Wu, S. F.: Bounds on the Performance of P2P Networks Using Tit-for-Tat Strategies, Seventh IEEE International Conference on Peer-to-Peer Computing, 2007

[Draves, 04] Draves S.: The Interpretation of Dreams, An Explanation of the Electric Sheep Distributed Screen-Saver, 37H37H37Hhttp://electricsheep.org/

[EGI, 10] European Grid Initiative funded by European Comission's Programme: http://web.eu-egi.eu

[Einstein, 10] Einstein@home project homepage: http://www.einsteinathome.org

[Foster, 03] Foster, I., Iamnitchi, A.: On Death, Taxes and Convergence of Peer-to-Peer and Grid Computing, Lecture Notes in Computer Science, Vol. 2735, 2003

[Fujimoto, 04] Fujimoto, N., Hagihara, K.: A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks, Proceedings of the 2004 International Symposium on Applications and the Internet Workshops (SAINTW'04), 2004

[Ganesan, 05] Ganesan, P., Seshadri, M.: On Cooperative Content Distribution and the Price of Barter, In Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), 2005

[Hei, 08] Xiaojun Hei, Yong Liu, K. W. Ross: IPTV over P2P streaming networks: the mesh-pull approach, IEEE Communications Magazine, Vol. 46, No. 2, 2008

[Hofmann, 05] Hofmann, M., Beaumont, L.: Content networking: architecture, protocols, and practice, Elsevier, 2005

[Liu, 08] Bo Liu, Yansheng Lu, Yi Cui, Yuan Xue: A measurement study on AS-aware P2P streaming strategies, Third International Conference on Communications and Networking in China (ChinaCom 2008), 2008

[Mathieu, 06] Mathieu, F., Reynier, J.: Missing Piece Issue and Upload Strategies in Flashcrowds and P2P-assisted Filesharing, Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, 2006

[Metz, 02] Metz, C.: IP anycast point-to-(any) point communication, IEEE Internet Computing, Vol. 6, No. 2, 2002

[Moca, 10] Moca M., Silaghi G.: Decision Models for Resource Aggregation in Peer-to-Peer Architectures, Grids, P2P and Services Computing 2010, 105-117, DOI: 10.1007/978-1-4419-6794-7_9

[Moreira, 08] Moreira, J., Antonello, R., Fernandes, S., Kamienski, C., Sadok, D.: A step towards understanding Joost IPTV, Network Operations and Management Symposium (NOMS), 2008

[Nabrzyski, 04] Nabrzyski, J., Schopf, J., Węglarz J.: (eds), Grid resource management: state of the art and future trends, Kluwer Academic Publishers, Boston, 2004

[OPNET, 10] Webpage of OPNET project: http://www.opnet.com

[Pióro, 04] Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufman Publishers 2004

[Rabinovich, 98] Rabinovich, M.: Issues in Web Content Replication, Data Engineering Bulletin, Vol. 21, No. 4, 1998

[Sarat, 06] Sarat, S., Pappas, V., Terzis, A.: On the Use of anycast in DNS, Proceedings of 15th International Conference on Computer Communications and Networks (ICCCN), 2006

[Schlosser, 06] Schlosser, D., Hobfeld, T., Tutschku, K.: Comparison of Robust Cooperation Strategies for P2P Content Distribution Networks with Multiple Source Download, Sixth IEEE International Conference on Peer-to-Peer Computing, 2006

[Shen, 09] Shen, X., Yu, H., Buford J., Akon, M., (eds.): Handbook of Peer-to-Peer Networking, Springer 2009

[Steinmetz, 05] Steinmetz, R., Wehrle K., (eds.): Peer-to-Peer Systems and Applications, Lecture Notes in Computer Science, Vol. 3485, 2005

[Subramanian, 05] Subramanian, R., Goodman, B.: Peer to Peer Computing: The Evolution Of A Disruptive Technology, Idea Group Publishing, 2005

[Tarkoma, 10] Tarkoma, S.: Overlay Networks: Toward Information Networking, Auerbach Publications, 2010

[Taylor, 03] Taylor I., Shields M., Wang I., Philp R.: Distributed P2P Computing within Triana: A Galaxy Visualization Test Case, Proceedings of the 17th International Symposium on Parallel and Distributed Processing (IPDPS), 2003

[Tjoa, 05] Tjoa, A. M., Andjomshoaa, A., Shayeganfar1, F., Wagner, R.: Semantic Web Challenges and New Requirements, Proceedings of the 16th International Workshop on Database and Expert Systems Applications, 2005

[Travostino, 06] Travostino, F., Mambretti, J., Karmous Edwards, G.: Grid Networks Enabling grids with advanced communication technology, Wiley, 2006

[Uppluri, 05] Uppuluri P., Jabisetti N., Joshi U., Lee Y., P2P Grid: Service Oriented Framework for Distributed Resource Management, IEEE International Conference on Services Computing – IEEESCC, 2005

[Vanderster, 09] Vanderster, D. C., Dimopoulos, N. J., Parra-Hernandez, R., Sobie, R. J.: Resource allocation on computational grids using a utility model and the knapsack problem, Elsevier Future Generation Computer Systems 25, 2009

[Wal06] Walkowiak, K.: Lagrangean Heuristic for Anycast Flow Assignment in Connection-Oriented Networks, Lecture Notes in Computer Science, Vol. 3991, Springer Verlag, 2006, 618-625

[Wilkinson, 09] Wilkinson, B.: Grid Computing: Techniques and Applications, Chapman & Hall/CRC Computational Science, 2009

[Wu, 05] Wu, G., Tzi-Cker, C.: Peer to Peer File Download and Streaming, RPE report, TR-185, 2005

[Wyrzykowski, 05] Wyrzykowski, R., Meyer, N., Stroinski, M.: CLUSTERIX: National Cluster of Linux Systems, Proceedings of International Conference on Linux Clusters, 2005

[Yang, 04] Xiangying Yang, De Veciana, G.: Service Capacity of Peer to Peer Networks, In Proceedings of INFOCOM '04, Vol. 4, 2004

[Zhang, 04] Li Zhang, Jia Weijia, Wei Yan, Xiao-Ming Li: An efficient anycast routing protocol based on multi-metrics, Proceedings of 7th International Symposium on Parallel Architectures, Algorithms and Networks, 2004