Journal of Network and Computer Applications 34 (2011) 327–340

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

An ant-based rate allocation algorithm for media streaming in peer to peer networks: Extension to multiple sessions and dynamic networks

Hadi Goudarzi [a,b], Amir Hesam Salavati [a,b,*], Mohammad Reza Pakravan [a,b]

[a] Data Communications Research Lab, Advanced Communication Research Institute, Faculty of Electrical Engineering, Sharif University of Technology, Tehran, Iran
[b] School of Electrical Engineering, Sharif University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 6 September 2009
Received in revised form
8 June 2010
Accepted 28 June 2010

Keywords:
Ant-colony optimization
Multimedia streaming
Peer to peer media streaming
P2P networks
Rate allocation

ABSTRACT

In this paper, we introduce a novel algorithm for rate allocation in media streaming P2P networks where multimedia contents are distributed among network members and streamed toward any requesting peer. The proposed algorithm is based on ant-colony optimization. It is capable of handling network dynamism, which is an inherent property of unstructured P2P networks. Another advantage of our algorithm is its ability to get over uncertainties in network state information, particularly the rate of supplying peers that could happen due to lack of accurate measurements. In addition, the suggested method does not rely on any information about the topology of the network.

We have investigated both single and multiple streaming sessions scenarios in which more than one peer is receiving media streams from media providers. We show that the suggested algorithm will reach the maximum achievable rate of the network quite fast. A key feature of the proposed algorithm is its low pass filter property, which makes it discriminate between transient and permanent network changes. If the changes are transient, the algorithm easily and rapidly compensates the temporary losses. In cases where the network changes last longer, the algorithm overcomes losses by employing other nodes that have the media stream available. The rate of adaptation is adjustable and must be carefully determined according to network conditions. Moreover, adaption rate is not constant and varies during the streaming session. This results in uninterrupted services for current users in cases where multiple sessions are present in the network.

Finally, since we have assumed that fountain codes are used to encode media streams in the P2P networks, the suggested algorithm does not require the user to receive different parts of the streams according to a predefined order and from a specific list of media suppliers. It suffices that the user gets as many stream chunks as necessary, regardless of their order or the fact that not all the media suppliers have all the parts available. In other words, using fountain codes enables us to overcome a big difficulty of P2P media streaming and that is to receive different parts of media streams according to a specific order.

1. Introduction

Peer to peer networks have grown rapidly in recent years. Although P2P networks were originally designed for file sharing purposes, due to their inherent properties, they are perfectly suitable for other distributed network-wide activities such media streaming and voice communication. As a result of improved users access bandwidth in the past decade, numerous commercial softwares are currently available for each of these purposes such as Skype (2009) for voice communications, Gnutella (2001) for content sharing, and SoPcast (2009) for video streaming.

The major difference between a general and a media streaming peer to peer system lies in the streaming process. While in a general P2P network users must download a file completely and then view it, in a media streaming architecture users view the media stream on the fly. The streaming bears some challenges to the network due to its real time nature and large bandwidth consumption. In addition, dynamic nature of peer to peer networks makes the streaming process even more difficult.

Since required bandwidth for receiving media streams is quite high and the upload bandwidth of users are limited, a single user usually cannot supply the necessary streaming rate for other peers. Therefore, in practice a subset of peers stream the media file to the requesting peers. Upon completion of the streaming session, the requesting peers become media suppliers. The sum

* Corresponding author at: Data Communications Research Lab, Advanced Communication Research Institute, Faculty of Electrical Engineering, Sharif University of Technology, Tehran, Iran. Tel.: +98 2166165922.
E-mail addresses: h_goudarzi@ee.sharif.edu (H. Goudarzi), salavati@ee.sharif.edu (A.H. Salavati), pakravan@sharif.edu (M.R. Pakravan).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.06.015

rate of supplying peers must be higher than the required rate of the media stream. In this paper, we will describe an ant-based rate allocation method for such a P2P media streaming system.

Ant-based algorithms are based on a relatively new concept which is inspired by the collective foraging behavior of some ant species. Ants communicate indirectly by using a chemical substance called pheromone. There are several algorithms that model and exploit this behavior for solving different problems such as graph-based NP hard combinatorial optimization problems, e.g. the traveling salesman problem (for a quick review see Dorigo and Caro, 1999).

The method used by ants in searching for food is specially interesting for applications in communication networks because of their distributed manner and automatic adaption to environmental conditions. As a result, there have been many ant-based solutions for network issues such as routing (Caro et al., 2005; Gunes et al., 2002; Heissenbüttel and Braun, 2003; Hussein and Saadawi, 2003; Ohtaki et al., 2006), congestion control (Schoonderwoerd et al., 1996, 1997), multicast scenarios (Guoying et al., 2000) and energy consumption in ad hoc networks (Srisathapornphat and Shen, 2003). Baran and Sosa (2001) have also proposed an improvement over traditional AntNet routing algorithm (Caro and Dorigo, 1998) which makes it more suitable for dynamic networks such as P2P or wireless networks. Via simulations and implementations, the authors show that routing algorithms based on ant colony meta-heuristic could be employed in commercial networks. In a more recent approach, a bio-inspired algorithm based the collective behavior of ants is proposed for grid computing (Forestiero and Mastroianni, 2009). The suggested algorithm constructs a self-organized and decentralized structure in the overlay network for grid computations.

The foraging behavior of ants and content sharing in peer to peer networks are even more similar. In the first case, there is an ant colony where ants start their journey from and bring food back to. Initially, ants do not know the location of food sources. Their task is to search for food and find those sources that are easier to access and then bring back the food to the colony. Since food resources are limited, the best strategy is to find multiple food sources and supply the colony with all of them. Similarly, in peer to peer networks, there is a node requesting for a file or media stream. Since the node does not know which peer nodes could provide the requested file or stream, it has to do a search across the P2P network, find those peers and start receiving data from them. Due to the limited bandwidth, the best strategy for the requesting peer is to receive the desired content from multiple sources to increase the reliability and distribute traffic load in the network to avoid congestions. These similarities between the self-organizing behavior of ant colonies and self-organization in peer to peer networks have already been observed in Babaoglu et al. (2005). Since they do not require any global knowledge about the network, ant algorithms are qualified for peer-to-peer networks. For a nice animated overview of ants foraging behavior see Myrmedrome (2008).

In traditional media streaming methods, a stream is split into multiple chunks. Each chunk is numbered and in order to listen to or watch that stream uninterruptedly, one must receive all chunks according to their numbered order. This method bears its own challenges to any media streaming algorithm as it must find different parts, download them, put downloaded segments into buffer and finally arrange them according to their order.

Due to the difficulty of the aforementioned method and the fact that our proposed method is based on an ant-based approach, we use fountain codes (MacKay, 2005) as the means of coding and storing stream chunks. This approach also increases the similarities between peer to peer networks and ant colonies. Because in real ant colonies, ants search for food, regardless of its type or *order*. This is just like the way fountain codes work: to receive a stream, all one needs to do is to collect sufficient stream chunks and then decode them. The chunks numbers or their order does not matter. Therefore, by using fountain code, and its more practical and powerful successors such as LT (Luby, 2002) and Raptor codes (Shokrollahi, 2006), we can overcome a major difficulty in P2P media streaming and that is getting different parts of the media stream according to a predefined order. In traditional P2P streaming (and file sharing) methods, a user has to receive different segments according to their position in the original stream. Normally, different media suppliers have different parts of the stream. Hence, not only the user has to identify the media suppliers which have the stream available, but also to determine the parts each supplier can offer. As a result, the rate allocation and streaming algorithms become much more sophisticated.

However, if fountain codes are used, all that matters is to receive as many segments as necessary, which is slightly more than the file size. There is no need to worry about the fact that not all the media providers have all the stream segments available. In other words, all the user needs to know is the list of the nodes that can provide the media stream, do a rate allocation for these nodes and then receive as many stream chunks as necessary, from which ever media supplier it can.

Moreover, due to random linear coding used in fountain codes, some sort of robustness is achieved. Additionally, fountain codes do not have numerous practical issues of Multiple Description Coding (MDC) that we used in our previous work (Salavati et al., 2008).

In this paper we propose an algorithm for choosing supplying peers and doing rate allocation for these nodes based on the foraging behavior of ants. We assume that media streams are coded with fountain codes and each candidate supplying peer have some parts of the desired media stream. While the suggested algorithm is very simple, it is capable of reaching maximum achievable rate of media streaming in a P2P network. The algorithm does not need any information about network topology and can be executed in a distributed manner. In addition, even if for any reason the rate of supplying peers is not their actual streaming rate, the proposed algorithm works without any problem.

1.1. Related works

Peer to peer media streaming has been a hot research area in recent years. Many researchers have worked on different aspects of media streaming in P2P networks. One of the most important works on multimedia streaming in overlay networks is oStream which investigates different streaming methods in one multicast session application layer networks. In Cui et al. (2004), it is assumed that one peer could provide necessary data rate for at least one other user so that media streaming could be implemented as a multicast communication. Another paper that has examined P2P media streaming more deeply is Tran et al. (2004), in which an algorithm for building multicast trees in peer to peer networks is proposed. Due to network dynamism of peer to peer networks, the suggested algorithm, which is called ZIGZAG, entails multiple mechanisms to monitor and maintain multicast trees. While previous papers considered only the single source case, where one node could provide the necessary rate for streaming media contents to other peers, multisource streaming is discussed in some works like Itaya et al. (2005a, 2005b). In Itaya et al. (2005a), the authors presented a multi source streaming system where multiple sources transmit packets of multimedia content to the receiving node with some redundancy. This increases the reliability of media streaming in a network where packets can get lost and nodes may or may not be available during the desired streaming session. One of the important works that

considers multiple source media streaming is Hefeeda et al. (2005). It suggests a method to find the best set of nodes to provide media stream from all nodes that have a media content available. In this method, it is assumed that the rate of peer nodes are accurately measured and the topology is known by the users. Nodes run an optimization algorithm to maximize the expected rate of media stream. In addition, a simple method is proposed to do rate allocation and monitor the streaming rate of peer users in the network called PROMISE (Hefeeda et al., 2003). PROMISE tries to find the optimum set and allocate proper rate when there are more than one intermediate router between the peers. One of the disadvantage of PROMISE is the large amount of overhead caused by the optimization algorithm. IPROMISE is proposed (Firooz et al., 2009) to solve the first issue. In Firooz et al. (2007) an algorithm is proposed to solve the second problem by implementing a suboptimal algorithm which has less overhead. This suboptimal algorithm first sorts peer nodes based on their availability. Then it adds each node to the active source list based on network topology. It is shown that while this method is suboptimal compared to PROMISE, its final rate is very close to what PROMISE's optimal algorithm yields. Another approach to choose the best set adaptively is proposed in Mushtaq et al. (2006). InRejaie and Ortega (2003) authors have investigated methods of changing the quality of media stream for different users according to their network resources. To achieve this goal, authors have examined quality adaptation and layered media coding methods. While peer to peer networking has been explored by many researchers, the application of ant-based algorithms in P2P networks is rather a new concept.

Although the ant metaphor has been successfully applied to routing of data packets both in wireless and fixed networks, not much is yet done about its applications in peer to peer networks, at least compared to its application in wired and wireless networks. Nevertheless, there has been an increasing interest in the applications of ant colony meta-heuristic in P2P networks. Babaoglu et al. (2005) has done a comprehensive research about the applicability of biological processes to distributed environments, including a discussion of ant-based methods in context. SemAnt (Michlmayr et al., 2006) suggests an ant-based query routing algorithm for P2P networks. The experimental approaches show that the algorithm produces robust results and converges fast. The author has extended the algorithm to include strategies for self-adaptation to volatile networks where nodes may leave or join at any time. MUTE (2007) is an ant-based peer to peer file sharing application. Its performance is investigated in Ciglaric and Vidmar (2006). It considers the case of network dynamism and its effects on query routing performance. Anthill (Babaoglu et al., 2002) is an open source framework for the design, implementation, and evaluation of ant algorithms in peer-to-peer networks. In Dong-hong et al. (2007) authors introduce a hybrid ant-based search algorithm (HASA) for media streaming P2P networks. In their algorithm, there is only a single media stream with different segments. Segments are distributed among network nodes and each segment has a unique pheromone type. Ant-like agents are used to find different segments and bring them to the requesting peer.

All of mentioned efforts are dedicated to the design of query routing algorithms in peer to peer networks. To the best of our knowledge, this is the first work on ant-based rate allocation for media streaming in peer to peer networks.

1.2. Paper organization

The rest of the paper is organized as follows: In Section 2 we briefly review ant-based routing algorithms. We give a detailed explanation of the ant-based rate allocation algorithm in Section 3. Section 4 extends the proposed algorithms to the case of multiple sessions running at the same time in the network. Section 5 shows simulation results. Finally Section 6 concludes the paper and explains future works.

2. An introduction to ant-based routing algorithms

2.1. Foraging behavior of ants

Ants establish the shortest path between food and their nest in a fully distributed and autonomous fashion. Ants first wander to search for food. When they find a food source, they return to the nest while leaving a trail of chemical substance called pheromone on their way back. The pheromone attracts other ants and guide them to the food. The probability that the ants coming later choose a path is proportional to the amount of pheromone on the path. Although pheromone evaporates and decays, the following ants also leave additional pheromone and thus reinforce the path. If more ants travel over a path, they deposit more pheromone on it. Inversely, larger amounts of pheromone attract more ants. This behavior results in a positive feedback effect and good routes to food sources are rapidly formed. Since the number of ants that complete their journey to the food source in a given time is larger on a shorter path than on a longer path, a shorter path can accumulate more pheromone and attract more ants.

At last, the shortest of alternative paths is selected. Most of ants take this shortest way to the nearest food source. Longer paths are also maintained because some ants are attracted to them by remnants of pheromone.

2.2. Ant-based routing algorithm

Based on these principles, an ant-based routing algorithm for communication networks was first suggested by Caro and Dorigo (1998). In the proposed algorithm, each node has a sort of routing table called pheromone table (see Table 1). Every node has an entry (row in pheromone table) for each possible destination in the network, and every table has an entry for each neighboring node. The (i, j) member of the pheromone table indicates the pheromone intensity between the current node and its jth neighbor for destination i. Pheromone table determines the routing behavior of nodes. When an ant arrives at a node, it chooses its next hop based on pheromone intensities of pheromone table in a probabilistic manner. This is in contrast to deterministic routing of other routing algorithms. Eq. (1) (Dorigo and Caro, 1999) gives the probability with which an ant destined to node i, chooses node j as its next hop. Here, node j is one of the neighbors of node k where the ant is currently visiting. This is called random proportional transition rule:

$$p_k(i,j) = \begin{cases} \dfrac{\phi_{ij}}{\sum_{l \in \Im} \phi_{il}} & \text{if } i \in \Im, \\ 0 & \text{if } j \notin \Im. \end{cases} \quad (1)$$

Table 1
Pheromone table.

Destination	Neighbor			
	a_1	a_2	…	a_m
d_1	p(1,1)	p(1,2)	…	p(1,m)
d_2	p(2,1)	p(2,2)	…	p(2,m)
⋮	…	…	…	⋮
d_n	p(n,1)	p(n,2)	…	p(n,m)

In this equation, $\Im$ is the set of node k's neighbors that the ant has not visited yet and ϕ_{ij} is the (i, j) element of the pheromone table, indicating the pheromone intensity on link (k, j) for all ants destined to node i. Each ant saves the path it travels so that when the destination is found, it could find its way back to the source. In its path backward to source, the ant updates pheromone tables of all nodes it has visited in the forward path according to the defined rules. Eq. (2) indicates the updating rule:

$$\phi_{iD}(t+1) = (1-\alpha) * \phi_{iD}(t) + \Delta\phi. \quad (2)$$

Here, $\Delta\phi$ is the amount of pheromone added to the link and α indicates pheromone evaporation. The amount of $\Delta\phi$ depends on network parameters such as delay and congestion. At the beginning of the algorithm, all links are initialized with a little amount of pheromone, $\phi_{ij}(0)$. Subsequently, the pheromone is updated according to Eq. (2) in each iteration.

2.3. Why ant-based algorithms are appropriate for P2P networks?

The simple ant-colony optimization meta-heuristic shown in the previous section illustrates different reasons why this kind of algorithms could perform well in peer to peer networks. Ant-based routing algorithms have several properties that make them an appropriate choice for peer to peer networks. More specifically, They are:

- simple,
- robust,
- purely distributed,
- multipath, which is an important and desirable property in unstructured peer to peer networks, and
- remarkably adaptable to network changes. When the shortest path is accidentally broken, a longer alternative path eventually becomes the new shortest path.

Of course ant-based algorithms have some disadvantages as well. For instance one of the biggest disadvantages of ant-based routing algorithms arises in highly dynamic networks where a large number of control packets is needed. In such cases, ant-based algorithms need a large amount of overhead before they find the shortest path.

3. Ant-based rate allocation: single session case

In this section we provide a detailed explanation of the ant-based rate allocation algorithm for media streaming over peer to peer networks when there is a single streaming session running.

3.1. Problem formulation

We model the peer to peer network as a connected graph $G(V, E)$ where V is the set of nodes and E is the set of links. It is assumed that there are a number of nodes called supplying peers, that can provide media contents. Those nodes that are willing to receive a stream, send their requests to these supplying peers. We call these users as requesting peers. Supplying peers start streaming based on user requests and network conditions. In general, requesting peers do not know the address of supplying peers. As a result, they search for these nodes across the network. In this paper, we assume that the requesting peer has done the search process and already knows the address of media providers.

Since supplying peers may leave the P2P network or stop their supply process, it is a good idea that each requesting peer tries to find multiple suppliers and receives its stream from them. Assume that a node D requires a media stream with rate $\Re$. It has found a group T of supplying peers that have the media content available. Now the problem is how to allocate rate according to which each node sends the requested stream. In the simplest case, the set T contains only one node and this node has sufficient available bandwidth to support the required rate, $\Re$. In practice, such a node usually does not exist. Even if there is such a node, it soon becomes congested. Therefore, we usually have many nodes none of which could support the required rate alone. It is the requesting peer D that specifies the rate of streaming from each supplying peer. We suggest an ant-inspired algorithm to do this rate allocation.

3.2. Ant-inspired rate allocation

The requesting peer D assigns a pair (R, A) to each supplying peer where (R_i, A_i) represent the rate and availability of peer i, respectively. R_i is the rate that peer i *claims* it can support. The availability of peer i, represented by A_i, is a measure of time during which i is on. In other words, it indicates the percentage of time the i is up and running. A_i is also determined based on the claims of peer i. The set T contains all of the peers that have the requested stream available. In the first step, the requesting peer, D, initializes its pheromone table. Then according to its desirable rate, $\Re$, it generates a number of agents, which we call forward ants. Now we have a network in which the requesting peer has a direct link to each media supplier. By direct link we mean a TCP/IP connection in the overlay network.

Now the only thing to do is to assign pheromone to links.[1] Having done that, ants follow the links with more pheromone in a probabilistic manner. We add pheromone to links according to the rate and availability of the media provider at the end of the link. To be more precise, pheromone on link i is proportional to

$$\Delta\phi = A_i * R_i. \quad (3)$$

Assume that D has generated N forward ants. The number of ants is proportional to $\Re$, i.e. $N = \beta\Re$, where β is a constant. In other words, the number of forward ants that the requesting peer D generates is proportional to its required streaming rate. β should be chosen such that it is not too large or too small. If it is too large, we will generate large amounts of unnecessary additional traffic. On the other hand, if β is too small, the number of forward ants would not be enough to get a good estimate of network conditions. The exact value of β depends on the problem specifications, $\Re$ in particular. However, setting β to a value such that N is in the order of a few kilobytes is a reasonable choice.

Since forward ants choose their links probabilistically, in the first step the same number of ants choose each link. Each supplying peer receives a number of forward ants. The more ants a peer receives, the higher rate it is asked to provide. In response to receiving forward ants, the supplying peer generates some backward ants and sends them back to D. The number of backward ants generated by peer i is proportional to the rate that it could provide, i.e. R_i. Therefore, there is an *indirected* relationship between the number of forward and backward ants. The number of forward ants is based on the expectations of D. which depends on the claimed availability and rate of the

[1] Note that this is quite unorthodox compared to other applications of ant-colony algorithms. Because ant colony meta-heuristic is often used in situations that the topology of the network is unknown and there are multiple hops between the source and destination. Nevertheless, we are employing ant metaphor here to perform *rate allocation* in a network where there is direct link between the source and the destination. As we will show in the paper, using an ant-based algorithm has several advantages over other approaches for the same goal. Smooth adaptation to network dynamism and congestion, the ability of handling measurement inaccuracies and supporting multicast is some of these advantages.

supplying peers while the number of backward ants reflects the true conditions of the network and the path between each supplying peer to D.

Note that backward ants do not carry traffic themselves but they represent the rate by which traffic is transmitted to D. Upon receiving these backward ants, D updates its pheromone table according to Eq. (2), where $\Delta\phi$ is given by Eq. (3). In other words, backward ants update the pheromone table which is located in D.

It is clear that the pheromone values for those supplying peers that could provide higher rates during longer periods of time grow rapidly. In the next steps, forward ants choose the peers that have higher rates due to the pheromone update rule. In a few steps, forward ants find the best set of supplying peers that could provide the desired rate. Algorithms 1 and 2 indicate the pseudo-code for the rate allocation algorithm in requesting peer and the supplying nodes, respectively.

Algorithm 1. Rate allocation algorithm: requesting peer's part.

1. Initialize pheromone tables.
2. Fix β such that $N=\beta\Re$ is in the order of a few kilobytes.
2. Send N forward ants to the supplying peers listed in T.
3. Upon reception of backward ants from the supplying peer i, update the pheromone table according to Eqs. (2) and (3).
4. Repeat steps 2 and 3 until the end of streaming session.

Algorithm 2. Rate allocation algorithm: supplying peer's part.

1. Determine nominal (claimed) rate and availability, i.e. (R, A).
2. Upon arrival of forward ants, start streaming and send backward ants.
3. Repeat steps 2 until the end of streaming session.

3.3. Algorithm properties

The suggested algorithm has several desirable properties some of which are listed below:

- *Adaptability to network congestions*: The proposed algorithm is adaptable to congestion. To see this, assume that a node i has a high rate R_i and it is available any time so that A_i equals 1. Thus, the pheromone intensity for this node is so large that all of the forward ants choose this node with high probability. This will result in congestion on the path between D and i. Therefore, only a fraction of backward ants will be received by D and the pheromone increment on that link is less than what it could be if all backward ants were received by D. Due to pheromone evaporation, the excess pheromone on this link will evaporate and the pheromone value for this node decreases eventually. Therefore, fewer forward ants will choose this node in next steps and less pheromone will be added to the link. Finally, the network reaches stability when there is no congestion on the links. This could be verified with simulation results of this paper.
- *Smooth adaptation to network dynamism*: Another property of the proposed algorithm is that it smoothly adapts itself to network changes. As an example, assume that node i leaves the network. This means that it could not provide rate R_i, at least temporarily. As a result, it will not send backward ants to D. Therefore, no pheromone will be added to its link and fewer forward ants will be sent to this peer. Eventually, pheromone will evaporate and none of the forward ants would follow the link to this node. If in the middle of this process, node i comes back, it will generate backward ants and its pheromone will rise again. Thus, forward ants will consider i as a potential media provider again.
According to Eq. (1), when the pheromone value for one neighbor is reduced, the probability that this peer is chosen is reduced as well. Hence, the probability that other peers are chosen increases. Therefore, whenever a peer leaves or fails, other supplying peers compensate this failure in a smooth fashion. This proves the adaptability of the proposed algorithm to network dynamism which is specially important in peer to peer networks.
Note that if the rate loss is very severe, the current set of nodes in the list T may not be able to supply D with its required stream rate. In such a case, D has to redo the search phase to find additional supplying peers and add them to T. Another important situation is when a supplying node joins the network. While it might seem better to update the list T upon such events, we do not add the newly joined media provider to T because of additional complexity and overhead unless it is necessary to do due to rate losses. In other words, as long as the current set of nodes in T are able to cope with the rate requirements of the requesting peer D, we do not modify T and ignore the event of joining a media supplier to the network.
There is a tradeoff between the evaporation rate and the speed of adaptation to network changes. If the evaporation rate is high, the algorithm rapidly adapts itself to the changes in the network. But this changes may be temporary. Thus, having a high evaporation rate is not desirable in all cases. The coefficient α shown in Eq. (2), which determines evaporation rate, should be adjusted carefully to cover all these scenarios. This adjustment could be done by the user according to network conditions. If network changes frequently and these changes are not temporary, it is better to have a high evaporation rate. However, if the rate of change is relatively low and most of changes are temporary, it is better to increase α so the algorithm handles these changes smoothly without over reaction. We will discuss this issue in a further section.
- *The ability to handle measurements inaccuracies*: The suggested algorithm is able to handle inaccuracies in rate measurements. There could be cases where the actual rate of node i is not what it claims. This could happen either because node i's rate is not measured with high accuracy or there are bottlenecks in the physical path between node D and node i. In such cases, since the actual rate is less than what node i claims to be, the rate at which backward ants are received by D are less than what it should be. As a result, the amount of pheromone added to link (D, i) is less than its maximum. Thus, in next steps fewer forward ants are sent over this link and more forward ants are sent toward other peers. This means that the rate at which D receives media stream from node i is less than its announced rate, either due to congestion in physical path or inaccuracies in rate measurements. It is obvious that to overcome congestion as mentioned above, we do not need any information about network topology. This is a great advantage in comparison to algorithms that need to know network topology to handle congestion, like PROMISE (Hefeeda et al., 2005) or IPROMISE (Firooz et al., 2009).
- *Supporting multicast*: As we will see shortly, the suggested algorithm is able to support multicast scenarios very easily. The key idea here is that when a requesting node receives part of a media stream, it becomes a media supplier for those parts of the stream. As a result, whenever a new peer requests to view the media stream, the list of its supplying peers not only contains the original media providers, but also all other multicast members that have joined the session previously.

4. Extension of the algorithm to multiple sessions

In the previous section, we introduced an algorithm for rate allocation in P2P media streaming networks where a single streaming session is running. In this section, we will extend the proposed method to cases where multiple multicast sessions are present in the network at the same time.

4.1. Problem definition and notations

"Multiple sessions" refer to cases in which there are two or more streaming sessions concurrently running in the network. In such cases, there are more than one media stream recipients. These media receivers may or may not be independent. In other words, they may have requested different or the same streams. The later case will result in a multicast scenario. In this section, we consider both cases.

Denote requesting peers with $D_1,\ldots,D_M$ where M is the number of sessions. Each node i has a different rate requirement for its stream, shown by $\Re_i$. As in previous section, network is modeled by a connected graph $G(V, E)$. Each node in the network has some media streams available and will send them to other users upon their request. The nodes willing to receive a media stream, have to search for other nodes that have the required stream available. As in the single session case, we assume that the search phase is done and nodes have their list of supplying peers ready. We represent the media providers list of node i by T_i. Therefore, all remains to do is to perform a rate allocation algorithm to define the rates according to which destination nodes receive media streams from media providers. The suggested algorithm is explained in what follows.

4.2. Rate allocation algorithm

The suggested method is essentially the same as the one explained in the last section, except for the fact that it is now run by multiple requesting peers. We consider the case of independent receivers first. Basically, each node has a list of supplying peers and performs the ant-based rate allocation method independent of others. It is not hard to see that as long as there are no conflicts among supplying peers' resources, such as bandwidth or time, there will not be any problems. However, such a scenario is too simplistic. In real networks, it occurs quite frequently that two nodes have a common supplying peer in their lists. In these cases, conflicts will certainly arise and requesting peers must compete for the resources.

Another common scenario would happen when two different peers have completely different media providers but somewhere in the physical network, their streaming routes meet and a bottleneck is resulted. Hence, these requesting peers are not competing for resources of a common media provider, as in the previous case, but somewhere in the network there is a bottleneck that causes conflicts.

While the first example above is still not easy to solve in a distributed P2P network, the second one brings about even more serious issues. The only way to get over it is to have a full knowledge of the underlying topology as well as the information about real-time traffic passing over each link, which seems impossible. Nevertheless, since the proposed algorithm relies on neither topological nor real-time traffic information, it should not have much problem handling such cases. First of all, all other media streaming sessions seem as background traffic to a requesting peer. Because the suggested algorithm does the rate allocation based on the supplying peers availability, A, and available bandwidth, R, the effect of other sessions is taken care of in the available bandwidth metric.

Moreover, in the previous section we mentioned that the algorithm is capable of smoothly handling variations in available bandwidth due to network dynamism. Therefore, if multiple sessions are running in the network, the algorithm adapts itself to this dynamism. In addition, since the suggested algorithm does not need any topology-related information, underlying bottlenecks would not cause any problem. To make it more clear, remind that if there is a bottleneck in the network backward ants will not reach the requesting peer. Thus, pheromone will not be added to the route that has a bottleneck. As a result, other routes become dominating in a gradual and probabilistic matter. Hence, the rate allocation method automatically adjusts itself to such bottlenecks without even knowing what has caused the bottleneck.

The same discussion also applies to the multiple sessions scenarios. The key point here is to note that from each peer's viewpoint, nothing has changed from the single session's scenario. The only difference is that the available rates are reduced which is, as we discussed, handled by shifting traffic to other routes. This is specially simple in the proposed ant-based algorithm because of its inherent load-balancing ability.

Therefore, the suggested rate allocation algorithm can easily handle conflicts in supplying peers' resources as this situation is similar to the case of having a bottleneck somewhere in the streaming path. In this case, the bottleneck is located in the beginning of the path, the media provider. In such situations, the media supplier does its best to answer the requirements of the receivers, i.e. it sends as many stream packets as its output bandwidth allows it. This means that some of the receivers do not receive the stream from this supplier with their expected rate. Therefore, the amount of pheromone corresponding to this media provider will not increase as much as expected. As a result, the receiver(s) try to compensate their loss by increasing their required rate from other suppliers.

In the simulation results section, we will consider above scenarios where multiple sessions are running in the network and show that our arguments also holds in simulated networks.

4.3. Multicast scenarios

The previous subsection was dedicated to non-multicast scenarios where all media recipients were independent. However, what happens if some peer would like to receive the same media stream? For instance, in case of a live soccer match, there would be many peers willing to watch the game. We show that the proposed algorithm is able to easily handle such multicast scenarios as well.

To see why, recall that when a peer receives a media stream, it becomes a media supplier itself. Thus, it could stream that media to other peers upon their request. Having this in mind, it is not difficult to see how the suggested ant-based algorithm deal with multicast scenarios: in the beginning, some nodes receives a media stream from some supplying peers. Then, whenever a new peer requests the media, its media providers list not only contains the original media streamers, but also the nodes that have received the media previously.

Therefore, from the viewpoint of new nodes, other multicast members are the same as the original media suppliers. As a result, all they need to do is to run the rate allocation method for their own list, which contains other previously joined multicast members. The rate allocation algorithm is essentially the same as before: it determines the rates based on the availability and available rate of the media suppliers, i.e. according to $R_i \times A_i$. Thus, it allocates more rate to those suppliers with larger $R_i \times A_i$ values, whether they are original providers or other multicast members.

Nevertheless, since original providers *usually* have higher rates and availability, they will normally be asked to provider higher streaming rates than multicast members.

Note that while this might seem different from usual multicast scenarios in physical networks, where a multicast tree is built and optimized, it is completely consistent with the definition of multicast in P2P networks where nodes frequently join and leave the multicast sessions.

5. A discussion on evaporation rate

Since evaporation rate is a key feature of the suggested algorithm, we have dedicated a whole section to it. We specifically address two issues here: the first is the necessity to adjust evaporation rate automatically and the second is the time-variant behavior of the evaporation rate in each session.

5.1. Adjustability of evaporation rate

As we mentioned before, evaporation rate, α, is a powerful means of adaptation to network changes. In networks where dynamism is high, α must be large to let pheromone on obsolete paths evaporate very fast. As a result, algorithm will replace old out-of-order routes with new fresh one. However, in a network where changes occur rather slowly and most of the time they are temporary, α must have a small value. Because in these cases, although network is somehow dynamic, most of the changes are short time. Therefore, if we choose a large α, the algorithm misinterprets temporary variations of the network with permanent ones and completely discard a route whose performance might have been degraded for a while but is a good route in general. Nevertheless, having a small evaporation rate, the algorithm does not completely discard the path. It rather switches some of the traffic to other paths until the failed route is restored.

Therefore, it can be easily seen that a good performance is achieved if evaporation rate is automatically selected based on network conditions. Such a goal can be accomplished by means of simple network measurements. For instance, the availability of supplying peers, A_is, are a good benchmark of network dynamism. Small availability figures is equivalent to high network dynamism and vice versa. Another simple measure could be the number of time a node was forced to redo the search for media suppliers, which clearly represents the amount of network dynamism. Other methods are possible as well. What is important is not the method but the fact that evaporation rate must be adjusted automatically.

5.2. Time-varying of evaporation rate

Evaporation rate should not be constant over the period of the streaming session. It must be large at the beginning and gradually reduce to a smaller value. To see why, consider an example scenario in which there is a streaming session running while another session is initiated. Most desirably, we would like the second session not to interrupt the first one and have both sessions running at the same time. If evaporation rate is constant, there would be cases in which the first session is interrupted. For instance, if both nodes try to receive a stream from a common media provider, there is a chance that their requests exceed the resources of the supplying peer. Thus, the supplying peer cannot provide required services for both of them. Since we have assumed that nodes do not store any network states, such as the time a session has been started, the supplying peer will not discriminate between the two nodes. As a result, there is a probability that the service of the first node is interrupted in favor of providing service for the second node.

However, if, as we suggested, the evaporation rate is larger at the beginning of a session, when the second node encounters a bottleneck or an unanswered request from the supplying peer, it assumes a permanent failure and seeks other media providers. At the same time, since the first session has been active for a longer time, it has a smaller evaporation rate. Therefore, the first node considers the variations in its streaming rate temporary and would not discard the supplying peer. Instead, it momentarily shifts some of the traffic to other routes and at a later time, when the main route is restored, conditions will go back to normal. We will investigate such scenarios in the simulation results section.

6. Simulation results

In this section, we investigate the performance of the proposed algorithm. We first consider single session case and then discuss the multiple session results. In the former case, we compare the proposed algorithm with PROMISE (Hefeeda et al., 2005). To do so, we have simulated ant-based rate allocation over the same topology as the one used in Hefeeda et al. (2005) and evaluated its performance. We examine different scenarios to show the advantages of the ant-based rate allocation compared to other similar algorithms.

Furthermore, we have performed simulations over a number of random topologies to assess the properties of ant-based rate allocation, convergence time and tolerance to measurement errors in particular.

To evaluate the performance of the algorithm when multiple sessions are running in the network, we have conducted two sets of simulations. In the first scenario, we have considered a network with 26 nodes and known topology. Having a know topology helps us in artificially introduce bottlenecks in the network and assess the performance of the algorithm in such cases.

Besides running simulations over a known topology, we have also conducted simulations over random large-scale networks with 100 nodes. A number of such topologies were generated and the convergence time of the ant-based rate allocation algorithm in the presence of multiple streaming sessions was evaluated.

Before reviewing the results, a clarification is needed over the notion of time in the simulation results. In the following figures, we have normalized time axis by the intervals of forward ants generation. In other words, each time step in the following results represent a duration which depend on the number of forward ants, N. Suppose we have $\Re = 1$ Mbps and $N = 8$ kbps. Therefore, we send 1000 one-byte forward ants in a second. Since the pheromone table is updated upon the arrival of a few backward ants, denoted by ℓ ($\ell = 10$ in simulations), the update frequency is equivalent to $N/8\ell = \frac{8000}{80}$ times per second. Therefore, each step of the algorithm is equivalent to $8\ell/N$, which is 10 ms in this example. Therefore, the time axis is normalized by $8\ell/N$ so that each step is represented with one unit.

6.1. Single session case

In this section, we discuss simulation results for the case where only one streaming session is running in the network. Although it is a fairly simple scenario, it helps us demonstrate main properties of our algorithm in adapting to network conditions, such as node departure, link failures and bottlenecks, and measurement errors.

We first consider a small topology, the same as the one used in Hefeeda et al. (2005), to compare the performance of ant-based rate allocation with that of PROMISE (Hefeeda et al., 2005).

6.1.1. Simulations over a simple network

In the single session case, we compare the proposed algorithm with PROMISE (Hefeeda et al., 2005). We have simulated some of the scenarios on the same topology that was used in Hefeeda et al. (2005) to show how PROMISE chooses the best set of nodes in a peer to peer network according to their rate and availability. While PROMISE needs to know the topology completely to choose the best set of nodes, our algorithm does not use any topology related information to do rate allocation. This topology is shown in Fig. 1. It is assumed that we have 6 media suppliers. The numbers besides each node indicates its announced rate and its availability. The capacity of each bottleneck link is shown as well. In all parts, we assume that the node D has requested a media stream with the rate ($\Re$) of 1 Mbps. Ant-based rate allocation does not need to know any topology related information and exact available rate of each node to do rate allocation. As a result, the proposed algorithm is much less complex than other suggested algorithms. In addition it could easily handle the errors in claimed rates due to network congestion or lack of accurate measurements. Therefore, in the simulations we have assumed that the proposed algorithm does not know the topology of the network. It only has access to the list of media providers with their claimed rate and availability, (R_i, A_i).

We first show the smooth adaptation of the proposed algorithm to network changes, which is one of its important advantages over similar rate allocation methods. In a normal peer to peer media streaming network, there are usually not many nodes with high rate and long availability like servers, i.e. with $A_i=1$. Moreover, network topology is very dynamic and network changes quite often. Thus, any algorithm designed for peer to peer networks must cope with network dynamism. In this section, we consider scenarios in which changes occur in network and examine the response of our algorithm to these changes. We have run all simulations over the topology shown in Fig. 1. In a P2P network, rate of nodes vary to a great extent. These variations could be temporary, long lasting or permanent. Any algorithm which is to be used in peer to peer networks should handle both of these changes. This is one of difficult challenges that algorithm designers have to deal with. It is mentioned in Mushtaq et al. (2006) that a low pass filter should be applied to network changes so that temporary variations is separated from permanent ones. From one perspective, ant-based algorithms implement a natural low pass filter. Since ants react to changes gradually, they completely discard a solution only if the duration of change is long enough. During this period, all of the pheromones for the invalid solution are completely evaporated. The rate of adaptation could be adjusted by carefully determining the evaporation rate of pheromone. This parameter could be chosen according to network conditions.

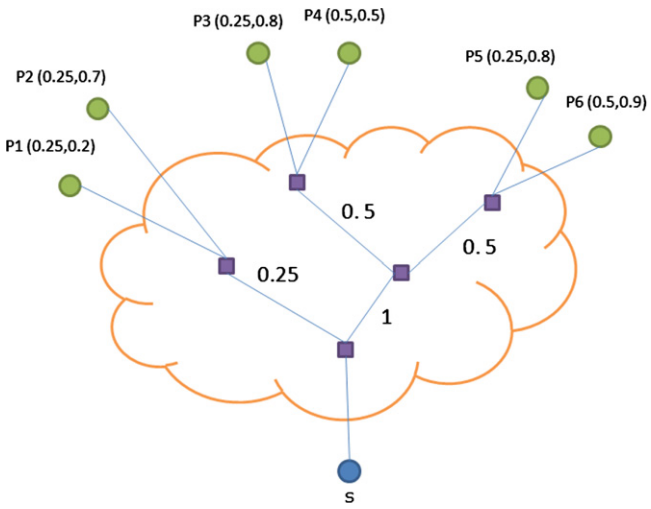

Fig. 1. Network topology.

To clarify this, consider the case shown in Fig. 2. All of the conditions are the same as before. At $t=500$, P_2 loses 50 kbps of its rate. As a result, the amount of pheromone for this link will not increase as much as before. As we discussed in previous sections and from Eq. (1), the probability that this node is chosen by forward ants will be reduced and other nodes should compensate this loss of rate. However, since all nodes are sending media streams up to their rate limits, this loss could not be compensated. Therefore, the pheromone of other users will not increase (Fig. 2). At $t=600$, network is recovered. If D had considered this variations as a permanent one, it would have added another node to the media providers set. It is clear that it does not add another node because this change is transient. When P_6 goes back to its normal conditions, gradually the pheromone of its link is increased and the rate of media stream rises to its required rate. The rate of pheromone evaporation must be adjusted such that the pheromone is not evaporated in 100 time units, which is the duration of variations. In the simulated case, α is set to 0.01. In other cases, it is set to 0.1.

Now assume another case where P_3 fails permanently. This could occur if P_3 leaves the network for example. In this case, maximum achievable rate decreases to 0.75 Mbps. Since the failure lasts for a long time, all of the pheromone on link to P_3 will finally evaporate and D will add another node, P_4, to the set of media providers. Since the initial amount of pheromone on the link toward P_4 is large, the rate of media stream from P_4 will be high. Therefore, some of the rate requirements from P_6 is assigned to P_4 and network converges to the stable solution. In Fig. 3 we have shown pheromone intensities for P_3, P_4 and, P_6.

In Fig. 3, the maximum achievable rate is 1.25 Mbps and we use only 1 Mbps of it. Now assume that P_2 loses 125 kbps of its rate permanently. Since the sum rate of nodes is still higher than 1 Mbps, there is no need to add a new node and the rest of the nodes compensate the loss of rate. This is because of the fact that when P_2 reduces its rate, its pheromone will decrease gradually due to pheromone evaporation. According to Eq. (1), the loss of pheromone of one node will decrease the probability that this node is chosen by forward ants and will increase the probability of choosing other nodes as the destination of ants. As a result, the rate of media streaming from these nodes will increase while the rate of P_2 will decrease. In the simulated case (shown in Fig. 4), node P_6 will increase its rate. This will compensate the loss of rate and keep the media streaming rate at its desired value, 1 Mbps. Small variations of rate in Fig. 4 is because all the other nodes could compensate the loss of rate due to P_2 failure. But after a few steps, node P_6 is chosen to do the compensation and other nodes continue to send their media stream as before.

Finally, consider the network condition shown in Fig. 2 that has reached stability. In this case, the maximum achievable rate is 1 Mbps and there is no excess rate since the required rate by D is also 1 Mbps. Now assume that P_6, whose *actual* (and not measured) rate is 450 kbps, for some reason reduces its rate to 250 kbps. Fig. 5 illustrates the pheromone intensities for P_5 and P_6. It could be seen that while the total capacity of the network was 1 Mbps, rate reduction of 200 kbps by P_6 does not change the rate of media stream received by D. P_5 compensate this loss shortly after the rate reduction, since it was not working at its limits due to the bottleneck on the links. When P_6 decreases its rate, P_5 increases its own rate. This scenario indicates that network changes in peer to peer networks do not deteriorate the performance of the suggested algorithm in all cases. The suggested algorithm could easily adapts itself to such changes.

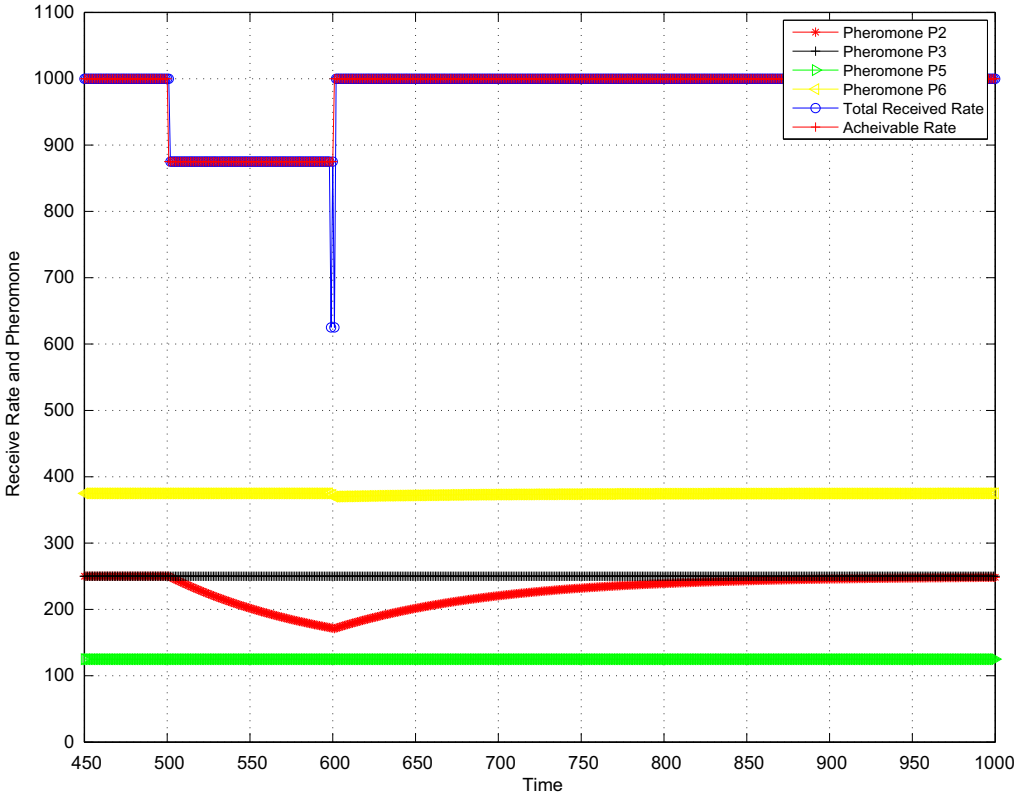

Fig. 2. Adaptation of rate allocation algorithm to transient rate variations.

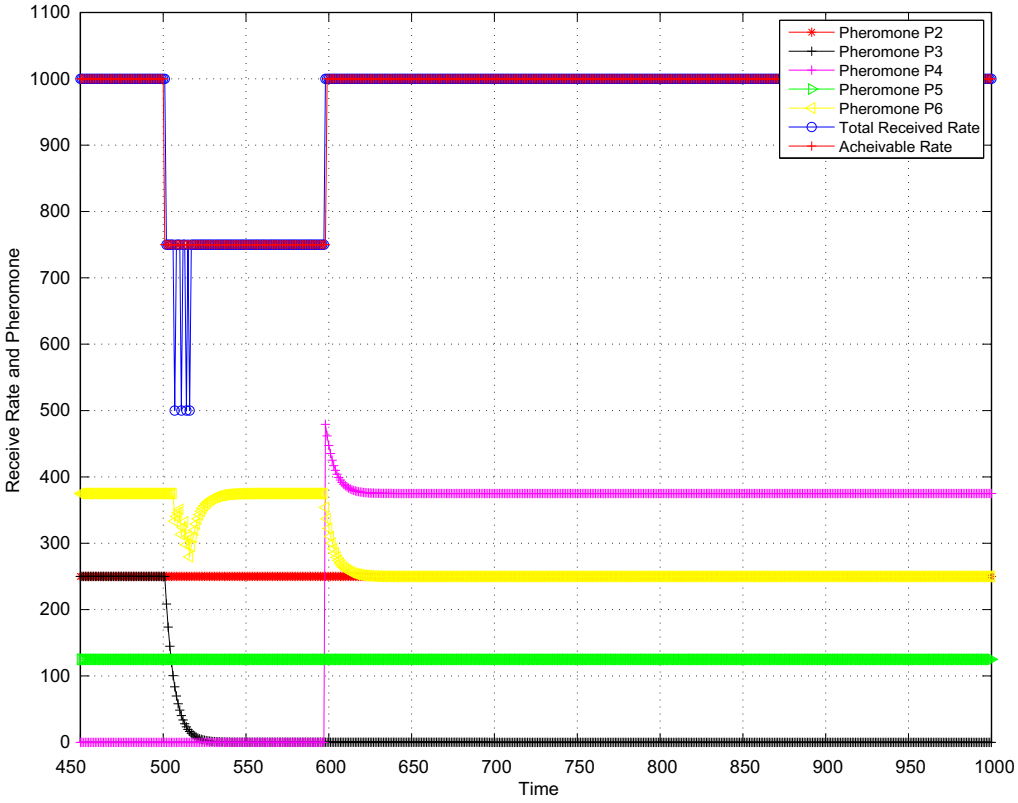

Fig. 3. Adaptation of rate allocation algorithm to permanent node failures.

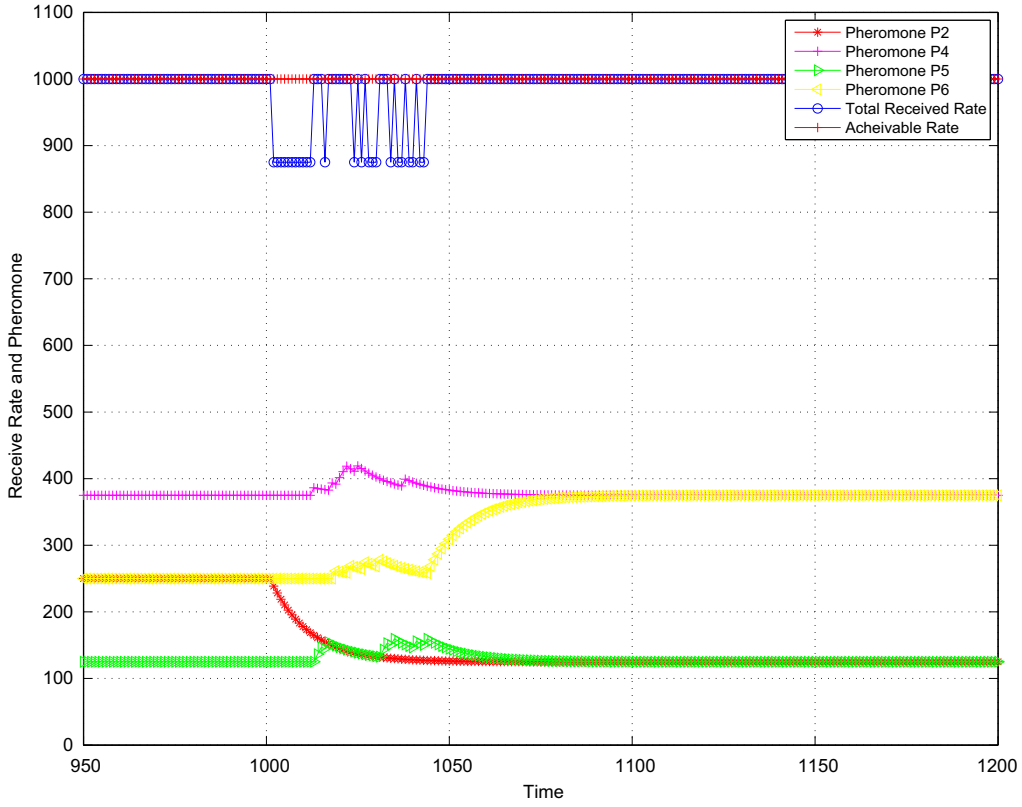

Fig. 4. Compensation of rate loss by other nodes.

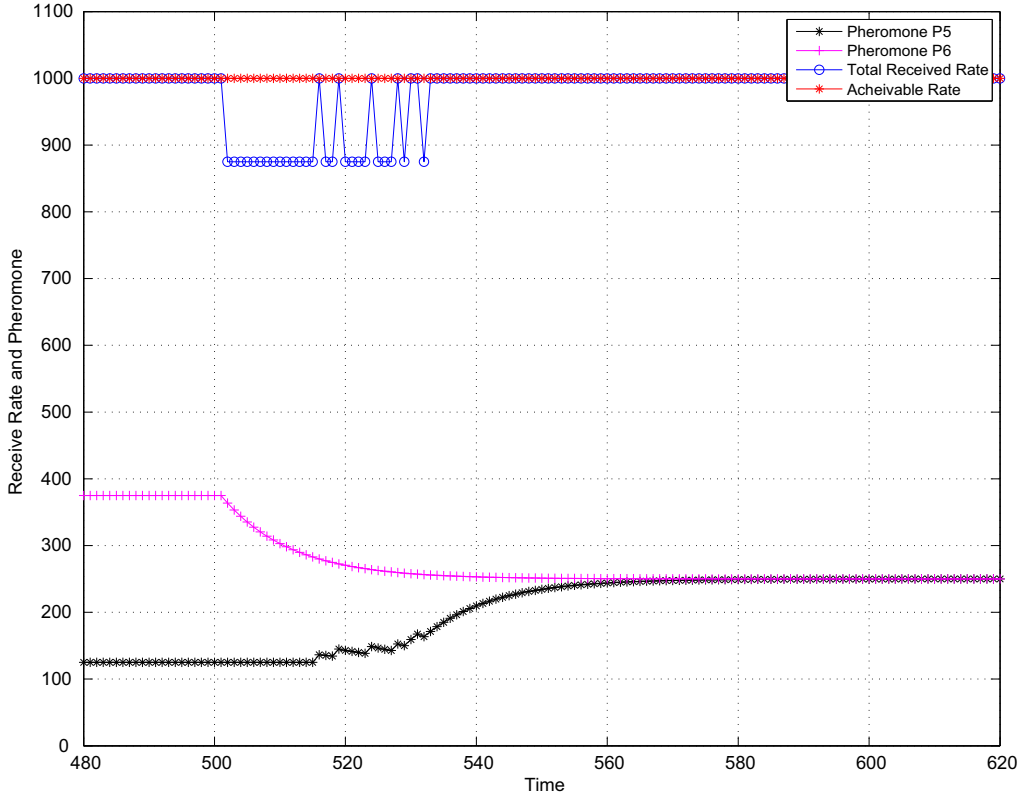

Fig. 5. Adaptation of rate allocation algorithm to permanent rate loss when there is a bottleneck in the network.

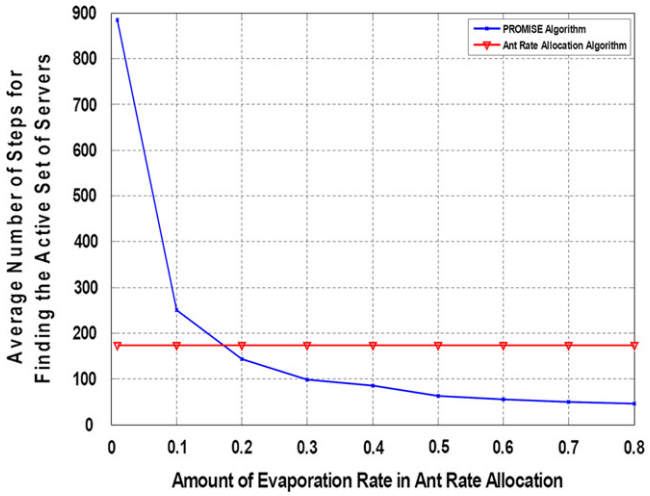

Fig. 6. Convergence time of the proposed algorithm compared to that of PROMISE (Hefeeda et al., 2003).

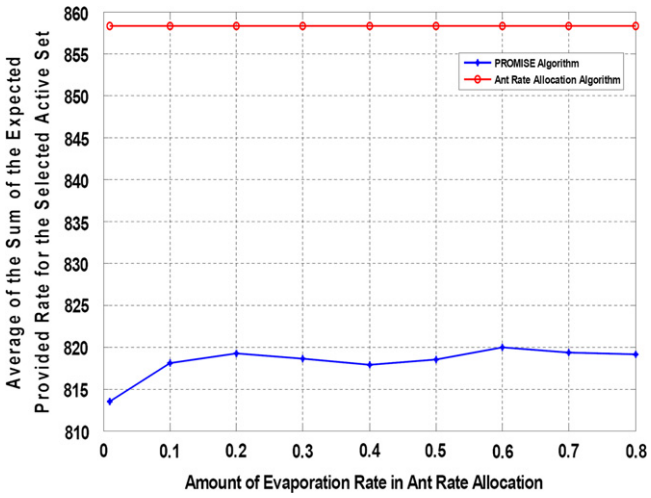

Fig. 7. Performance of the proposed algorithm compared to PROMISE (Hefeeda et al., 2003).

We note that the adaptation time of our algorithm can be varied by changing the parameter α and it can be determined by noticing the state of the nodes of P2P network.

6.1.2. Random topologies

In this section, we discuss the simulation results for random topologies with 15 nodes, 10 of which are peers with media streams and the other five are intermediate nodes in the networks. We generated 100 random topologies. Over each topology, the rate allocation algorithm is performed with different parameters, according to the criterion to be evaluated.

Fig. 6 depicts the average convergence time of the proposed algorithm versus evaporation rate, α. We also computed the average convergence time for PROMISE which is also shown in Fig. 6 as well.

It is clear that as evaporation rate increases, the algorithm converges faster. If α is greater than 0.18, the proposed algorithm converges faster than PROMISE. The higher the evaporation rate is, the less the convergence time of the algorithm will be. To transform the time unit in the above figure to actual time units, we have to follow the same technique as mentioned earlier in this sections. We have that $\Re = 1$ Mbps, $N = 8$ kbps and $\ell = 10$. Therefore, each step of the algorithm, and the time unit in Fig. 6, corresponds to 10 ms.

Fig. 7 illustrates the achievable rate of our algorithm compared to that of PROMISE (Hefeeda et al., 2003) for different evaporation rates. While the achievable rate of our algorithm is clearly less than PROMISE, the difference is not very much (PROMISE achieves about 7% higher streaming rates). Meanwhile, from this figure it is clear the performance of the proposed algorithm is not degraded when evaporation rate is increased. Therefore, from Figs. 6 and 7 if we increase evaporation rate, the suggested algorithm converges faster while maintaining an acceptable streaming rate compared to PROMISE.

The next evaluation criterion is tolerance to measurement inaccuracies. As we mentioned before, the suggested algorithm is able to handle inaccuracies in network state information such as peers availability and self-reported streaming rate. To investigate this capability of the proposed method, we created 100 random topologies. In each topology, we introduced a random amount of error with specified mean to supplying peers streaming rates, R. In practice, this error may be caused either because of congestion or measurement inaccuracies. Fig. 8 illustrated the convergence time of the algorithm as a function of average error percentage in supplying peers' claimed streaming rates. It is clear that as the amount of error increases, the algorithm spends more time reaching the optimal condition. Nevertheless, Fig. 8 shows that the suggested algorithm is able to handle a fair amount of rate inaccuracies and adapt itself to errors in claimed streaming rates.

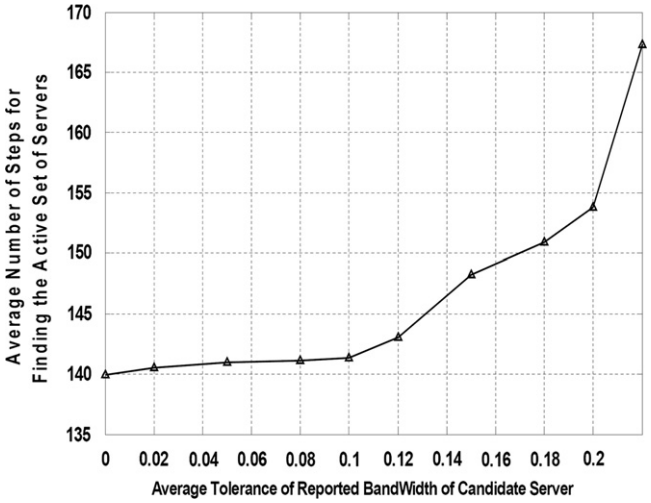

Fig. 8. Convergence time versus measurement errors in streaming rates.

6.2. Multiple sessions case

In this section, we investigate the performance of the suggested rate allocation algorithm when multiple streaming sessions are present in the network. We have considered cases in which a streaming session has been started for a while when the second session starts.

We first consider a network with 26 nodes and a know topology. Having the topology helps to introduce bottlenecks so that we are able to demonstrate properties of the suggested rate allocation algorithm. Of the 26 nodes of the considered network, 18 are streaming peers and eight are other intermediate nodes, as shown in Fig. 9. The available rate and the availability measure of each peer is also shown in the figure.

In the simulated scenario, the first session starts at the beginning of the simulation and the second session (for the second receiver) starts at $t = 500$. For the first session, in the first attempt (P_1, P_2, P_3, P_8, P_9) is selected. Because a bottleneck exists

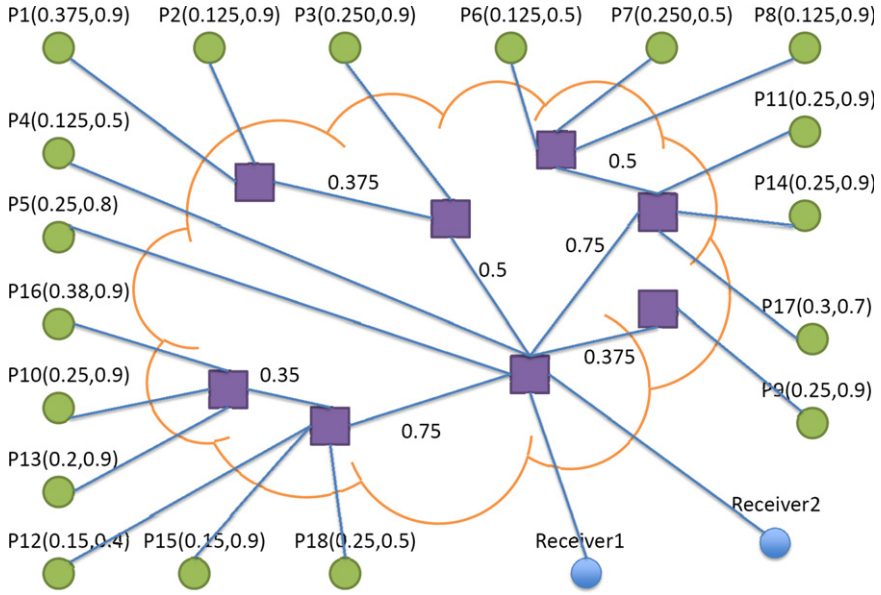

Fig. 9. Network topology for multiple sessions case.

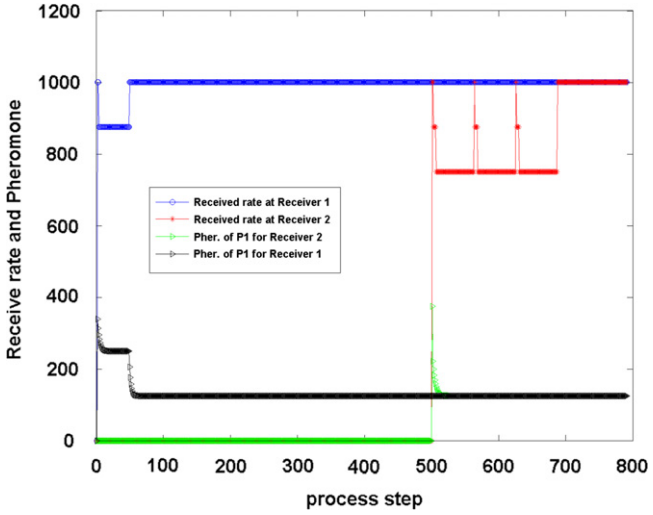

Fig. 10. The total received rate and the pheromone of P_1 for both sessions.

for P_1, P_2 and P_3, the sixth node P_{10} is added and the first receiver achieves the desired rate by this selection.

At $t=500$, the second session starts. In the first attempt it selects (P_1, P_{11}, P_{13}, P_{14}, P_{15}) but this selection has two conflicts with first session selection: P_1 (the whole remaining rate of P_1 is not achievable because of active session from P_1, P_2 and P_3 for first session) and P_{10}, (which has a bottleneck with P_{13}).

As explained in previous sections, the evaporation rate of streaming sessions decreases over time which helps them to maintain an interrupted service. Therefore, by the time the second session starts, the evaporation rate of the first session is reduced to a small amount. Therefore, the first session remains untouched and the second receiver finds out that the received rate is not the desired one. As a result, P_{13} is deleted form the selected transmitters. Moreover, based on the indirect feedback given by P_1 via backward ants, the rate from P_1 is decreased to possible rate of 0.125 Mbps. Fig. 10 illustrates the evolution of pheromone of P_1 for the two receivers over time. To compensate this gap between the required and actual rate, P_{16} is added to the active users.

However, P_{16} also has conflict with P_{10} from the first session and after a short time, this transmitter is deleted from the second session. Then P_{17} is added and because a bottleneck exists between P_{11} and P_{17}, one of them which is P_{11} is deleted from the selected set. Finally, P_5 is added to the second session. This completes the list. The total streaming rates of the two receivers are shown in Fig. 10.

Therefore, in the simulated scenario, we have bottleneck with a previously established connection and even an intermediate node. Nevertheless, because of decreasing the evaporation rate of the previously established connection, this session is not interrupted and the second session adapts itself to network conditions.

Table 2
Convergence time of the algorithm for two streaming sessions over random networks.

	First receiver	Second receiver
Convergence time (s)	2.33	2.18

6.2.1. *Random topologies*

While simulations over a known topologies helps us in clarifying the properties of the algorithm, to evaluate practical aspects of the algorithm we have also simulated the method over large-scale random networks. The considered topologies are composed of 90 nodes, 50 of which are streaming peers and the other 40 are intermediate nodes. Availability and rate of each peer was generated according to uniform probability distribution between [0, 1] and [0.05, 0.25] Mbps, respectively. Furthermore, the maximum achievable rate for each connection of intermediate nodes (which can be assumed that they are the bottleneck points) are generated randomly based on a uniform distribution between [0.2, 0.3] Mbps.

We considered the scenario in which two streaming sessions are running, the first one starts at the beginning of the simulation and requires a 1 Mbps streaming rate. The second session starts at $t=5$ s. It also requires a rate equal to 1 Mbps. We generated 1000 random topologies with mentioned specifications and performed

this scenario over them to determine the convergence time of the algorithm.[2] Table 2 indicate the results.

Therefore, the average number of steps for convergence for both sessions is in the same order. This shows that the number of sessions does not affect the convergence time of the proposed algorithm as long as there are enough resources for all the streaming sessions. Any conflicts and bottlenecks are automatically handled by the algorithm in a smooth fashion.

In only 5% of the cases, the established connection between media providers and the receiver in first session is affected when the second streaming session starts. This is acceptable because if most of the providers of the second session is selected from the active media suppliers of the first session, it is possible that lowering the evaporation rate for established connection is not enough and the newly established session affect the connection of the first session.

7. Conclusions and future works

In this paper, we proposed a new bio-inspired algorithm based on ant-colony optimization meta-heuristic for performing rate allocation in media streaming peer to peer networks. The suggested algorithm is completely distributed and able to adapt to inherent dynamism of unstructured P2P networks. Moreover, it does not need any topology related information and can overcome inaccuracies and uncertainties in estimated network state information such as supplying peers' streaming rates. In addition, due to the assumption that fountain codes are used to encode media streams in the P2P networks, our algorithm does not require the user to receive different parts of the streams according to a predefined order and from a specific list of media suppliers. All a user needs to do is to determine the list of media suppliers, perform the rate allocation algorithm over this list and start receiving stream chunks from these nodes with the determined rate, regardless of the fact that not all these nodes contain all the stream segments.

Another desirable property of the suggested approach is its low-pass filter characteristic which help it discriminate between transient and permanent network changes. If the changes in the network are transient, the algorithm compensates the temporary losses without much complexity. In case of long term network changes such as failure of links or nodes, the algorithm overcomes losses by employing other supplying peers from its list of media providers. The rate of adaptation could be adjusted according to network conditions.

We have extended the algorithm to cases where multiple streaming sessions are running in the network and showed that the proposed approach could support overlay multicast as well. In addition, because of time-variant evaporation rate, the suggested algorithm could provide uninterrupted streaming services for users.

We demonstrated that the suggested rate allocation algorithm will reach the maximum achievable rate of the network quite fast and with relatively little overhead.

We can further improve the performance of the algorithm by considering the possibility of updating the list during the streaming process so that we have the best list of media providers with the least possible cost. This is a subject of our future research.

Another way of improving the algorithm is to optimize the evaporation rate, α, according to network dynamism. As mentioned in the paper, adapting evaporation rate to network conditions helps the algorithm to converge faster. In the simulations, a very simple scheme of selecting between two values ($\alpha = 0.1$ and 0.01) for dynamic and static networks is used. Nevertheless, defining a measure for network dynamism and relating to a carefully chosen function of this measure will certainly improve the performance.

Our future works will also include designing an ant-based search algorithm to find supplying peers across the P2P network. Implementing the ant-based rate allocation algorithm in a real world test-bed will be another important aspect that can be pursued in future.

[2] To calculate the convergence time from the number of steps required by the algorithm to converge, we followed the same approach as mentioned in the beginning of this section by noting that $\Re = 1$ Mbps, $N = 8$ kbps and $\ell = 10$.

Acknowledgments

We would like to specially thank Mr. Pouya Shariat Panahi because of his helpful ideas and opinions. We would also like to thank anonymous reviewers who have helped us improving the quality of this paper with their comments and opinions.

References

Babaoglu O, Meling H, Montresor A. Anthill: a framework for the development of agent-based peer-to-peer systems. In: Proceedings of the international conference on distributed computing systems, 2002. p. 15–22.

Babaoglu O, Canright G, Deutsch A, Caro GD, Ducatelle F, Gambardella LM, et al. Design patterns from biology for distributed computing. In: Proceedings of the European conference on complex systems, 2005. p. 26–66.

Baran B, Sosa R. AntNet: routing algorithm for data networks based on mobile agents. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 2001;5(12):75–84.

Caro GD, Dorigo M. AntNet: distributed stigmergy control for communications networks. Journal of Artificial Intelligence Research 1998;9:317–65.

Caro GD, Ducatelle F, Gambardella LM. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transactions on Telecommunications 2005;16:443–55.

Ciglaric M, Vidmar T. Ant-inspired query routing performance in dynamic peer-to-peer networks. In: Proceedings of the 20th IEEE international parallel and distributed processing symposium, 2006. p. 25–9.

Cui Y, Li B, Nahrstedt K. oStream: asynchronous streaming multicast. IEEE Journal on Selected Areas in Communications 2004;22(1):91–106.

Dong-hong Z, Xu D, Zong-kai Y. Hybrid ants-like search algorithms for P2P media streaming distribution in ad hoc networks. Journal of Zhejiang University Science A 2007;8:1191–8.

Dorigo M, Caro GD. The ant colony optimization meta-heuristic. In: Corne D, Dorigo M, Glover F, Dasgupta D, Moscato P, Poli R, editors. New ideas in optimization. McGraw-Hill; 1999. p. 11–32.

Firooz MH, Ronasi K, Pakravan MR, Avanaki AN. A multi-sender multicast algorithm for media streaming on peer-to-peer networks. Computer Communication 2007;30(10):2191–200.

Firooz MH, Avanaki AN, Pakravan MR, Ronasi K. A fast and reliable multi-sender algorithm for peer-to-peer networks. Journal of Network and Computer Applications 2009;32(3):733–40.

Forestiero A, Mastroianni C. A swarm algorithm for a self-structured P2P information system. IEEE Transactions on Evolutionary Computation 2009;13(4):681–94.

Gnutella 〈www.gnutellanews.com〉; 2001.

Gunes M, Sorges U, Bouazizi I. ARA—the ant-colony based routing algorithm for manets. In: Proceedings of the IEEE parallel processing workshops, 2002. p. 79–85.

Guoying L, Zemin L, Zheng Z. Multicast routing based on ant algorithm for delay-bounded and load-balancing traffic. In: Proceedings of the IEEE local computer networks (LCN 00), 2000. p. 362–8.

Hefeeda M, Habib A, Botev B, Xu D, Bhargava B. PROMISE: peer-to-peer media streaming using collectcast. In: Proceedings of the ACM multimedia (MM'03), 2003. p. 45–54.

Hefeeda M, Habib A, Xu D, Bhargava B, Botev B. CollectCast: a peer-to-peer service for media streaming. Multimedia Systems 2005;11:68–81.

Heissenbüttel M, Braun T. Ants-based routing in large scale mobile ad-hoc networks. In: Proceedings of the kommunikation in verteilten systemen (KiVS03), 2003. p. 91–9.

Hussein O, Saadawi T. Ant routing algorithm for mobile ad-hoc networks (ARAMA). In: Proceedings of the IEEE performance, computing, and communications conference, 2003. p. 281–90.

Itaya S, Enokido T, Takizawa M. A high performance multimedia streaming model on multi-source streaming approach in peer-to-peer networks. In: Proceedings of the advanced information networking and application, 2005a. p. 27–32.

Itaya S, Enokido T, Takizawa M. A scalable multimedia streaming based on multi-source streaming concept. In: Proceedings of the parallel and distributed systems, 2005b. p. 15–21.

Luby M. LT codes. In: Proceedings of the symposium on foundations of computer science, 2002. p. 271–80.
MacKay DJC. Fountain codes. IEE Proceedings Communications 2005;152(6):1062–8.
Michlmayr E, Pany A, Graf S. Applying ant-based multi-agent systems to query routing in distributed environments. In: Proceedings of the third IEEE conference on intelligent systems 2006 (IEEE IS'06), 2006. p. 36–41.
Mushtaq M, Ahmad T, Meddour DE. Adaptive packet video streaming over P2P networks. In: Proceedings of the scalable information systems, 2006.
MUTE: simple, anonymous file sharing ⟨www.mute-net.sourceforge.net⟩; 2007.
Myrmedrome: a real ant-colony simulator ⟨www.not-equal.eu/myrmedrome⟩; 2008.
Ohtaki Y, Wakamiya N, Murata M, Imase M. Scalable ant-based routing algorithm for ad-hoc networks. IEICE Transactions on Communications 2006;E89-B(4):1231–8.
Rejaie R, Ortega A. PALS: peer to peer adaptive layered streaming. In: Proceedings of the 13th international workshop on network and operating systems support for digital audio and video, 2003. p. 153–61.
Salavati AH, Goudarzi H, Pakravan MR. An ant based rate allocation algorithm for media streaming in peer to peer networks. In: Proceedings of the local computer networks (LCN 08), 2008. p. 456–63.
Schoonderwoerd R, Holland O, Bruten J, Rothkrantz L. Ant-based load balancing in telecommunications networks. Adaptive Behavior 1996;5(2):169–207.
Schoonderwoerd R, Holland O, Bruten J. Ant-like agents for load balancing in telecommunications networks. In: Proceedings of the first international conference on autonomous agents, 1997. p. 209–16.
Shokrollahi A. Raptor codes. IEEE Transactions on Information Theory 2006;52(6):2551–67.
Skype ⟨www.skype.com⟩; 2009.
SoPcast ⟨www.sopcast.org⟩; 2009.
Srisathapornphat C, Shen CC. Ant-based energy conservation for ad hoc networks. In: Proceedings of the international conference on computer communications and networks (ICCCN 03), 2003. p. 32–7.
Tran DA, Hua KA, Do TT. A peer-to-peer architecture for media streaming. IEEE Journal on Selected Areas in Communications 2004;22(1):121–33.