Load-Aware Self-Configuration in Mobile Peer-to-Peer Applications

Diego N. da Hora, Daniel F. Macedo, José Marcos S. Nogueira
Computer Science Department
Universidade Federal de Minas Gerais, Brazil
Emails: {dnhora,damacedo,jmarcos}@dcc.ufmg.br

***Abstract*—Peer-to-peer (P2P) applications must be configured for each scenario in which they are executed. The determination of the ideal configuration is costly, as it requires a complete characterization of the network. Thus, P2P networks usually rely on a generic configuration, which provides a reduced performance when compared to the best manual configuration. This work proposes two autonomic controllers that reconfigure the unstructured P2P protocol in run time, based on the conditions of the network. The controllers were evaluated on mobile networks employing the Gnutella protocol, however they can be deployed over any mobile P2P protocol based on flooding or gossiping. Simulation results show that the performance of the controllers is comparable to that of the best manual configuration, both in number of queries resolved as well as the response time.**

I. Introduction

Mobile ad hoc networks (MANETs) are frequently employed in rescue operations and in battle fields due to their independence to previous infrastructure [1]. Their characteristics preclude the use of client-server architectures, since servers become points of vulnerability. The Peer-to-Peer (P2P) [2] paradigm thus is more suitable on such situations, since it mitigates task overload by distributing them among nodes [1].

There are two main approaches for content query in P2P networks: structured approaches and unstructured approaches. In the structured approach, peers build distributed indexes, and hence queries are implemented as a lookup operation in those indexes. Unstructured networks, on the other hand, do not have such an index, and as a consequence peers have to contact all peers to identify which of them have the information sought. Previous work characterized the performance of those two approaches in a MANET context [3]. Although structured networks are known in the literature for their outstanding performance over wired networks, unstructured networks outperform structured networks in most wireless environments, due to node mobility and the unreliability of wireless links.

Queries in unstructured P2P protocols use flooding-like techniques (broadcast messages, gossiping or others) in order to reach several neighbors at a time. This characteristic provides a high degree of tolerance to peer and link failures. However, flooding reduces the scalability of those protocols, such that network congestion caused by too many queries is the root cause for a degraded performance on large P2P networks in wireless deployments [3]. To cope with this problem, one may adjust the parameters of the P2P application to reduce the amount of query messages sent.

The parameters of unstructured P2P protocols should be adjusted according to the characteristics of each deployment. Usually, there is a trade-off between coverage and redundancy, measured by the amount of nodes contacted on each query, and the network load, measured by the amount of messages sent per unit of time. As an example, in the Gnutella protocol the number of neighbors as well as the time-to-live (TTL) of the queries are configurable parameters. Larger values of TTL and number of neighbors increase the coverage, potentially providing more content sources. Further, since the query will pass through more nodes and more paths, it will be more resilient to failures. However, a higher coverage induces more messages, which may increase the load of the network beyond peak capacity. Thus, the overall performance of the application degrades due to collisions and delays induced by queueing.

The manual adjustment of the parameters of a mobile P2P application requires a skilled practitioner and may need information about the amount of background traffic, rate of link failure and others. As a consequence, the determination of the best configuration is subject to errors and may require a significant amount of time.

This paper proposes that P2P protocols should auto-configure themselves in order to avoid costly error-prone human intervention. In order to achieve this, the paper proposes two autonomic controllers that re-configure unstructured P2P networks at run-time. The controller, tailored to MANET deployments, adjusts the parameters of the P2P application to cope with changes in network conditions. The proposed controllers act on the TTL of queries, increasing it when congestion is low, and reducing it when the network becomes congested. One controller employs a congestion control mechanism similar to TCP's, while another employs machine learning techniques to identify congestions. The controllers were evaluated via simulations in a scenario with varying network load. Simulations showed that the solutions consistently perform as well as the best performing static configuration. Although the controller were evaluated over a mobile implementation of Gnutella, they are applicable to any unstructured or hybrid mobile P2P protocol.

The remainder of the paper is organized as follows. Section II shows the related work. Section III presents the proposed autonomic controllers while SectionIV presents their tuning. Section V describes the simulation setup and the results. Finally, Section VI shows the conclusions and future work.

978-3-901882-50-0

II. Related Work

The integration of MANETs and P2P networks has been extensively studied [1]. Oliveira et al. evaluated the performance of an unstructured P2P network over different routing protocols (DSR, AODV, DSDV) [4]. Franciscani et al. proposed re-configuration algorithms that adjust the topology of the P2P network to resemble the physical topology of the underlying MANET [5]. Conti et al. proposed cross-layer optimizations to Gnutella running over MANETs [6].

Others proposed a hybrid approach, where communities of similar interests are created, and search is performed only within those communities [7, 8]. An index is created for the communities, however search within them is still performed using an unstructured approach.

Adaptive P2P networks were proposed to reduce the Internet traffic caused by heavy P2P use. Jiang and Jin propose a two-phase protocol where the TTL of the queries is adjusted according to the popularity of the information being requested [9]. The ideal TTL is determined by sampling a small part of the network using a query with a reduced TTL, then the query is restarted with the calculated TTL. Since both works are aimed at wired networks, they adapt the value of the queries to metrics such as the file's or node's popularity. Choffnes and Bustamante propose a neighbor selection policy that identifies the peers on the same AS, adapting the topology to reduce inter-ISP traffic [10]. A hybrid system that adapts the search technique based on the query terms was proposed in [11].

In mobile networks, on the other hand, the P2P application must be aware of the uncertainties and frequent packet losses of the wireless links as well as the highly dynamic physical topologies. ORION is a context-aware mobile P2P application that downloads files from nodes having more stable routes, in order to avoid disconnections [12]. The Cimbyosis platform employs selective caching in order to reduce the communication overhead [13]. Hara and Madria investigated replication strategies to ensure consistency among replicas [14].

We identified in our previous work [3] that the high amount of query packets produced by unstructured protocols generates collisions and packet losses, reducing the efficiency of the P2P application. On unstructured networks, on the other hand, replication is beneficial since those methods usually rely on a single query message, which frequently is lost in mobile networks. Based on those results, we proposed simple modifications to those types of P2P protocols in order to improve their performance over wireless links [3]. In [15] we defined a policy-based adaptation that switches from Gnutella to Random Walk query dissemination according to the load experienced on the network, in order to benefit from the advantages of each dissemination paradigm. The policies, however, need to be adapted for every deployment. Finally, we proposed a fuzzy controller to adapt the number of neighbors in unstructured P2P protocols [16], in an attempt to reduce the congestion caused by the P2P queries. This work advances the previously proposed algorithms since the devised controllers employ more elaborate control schemes.

GIA [17] proposes a flow control strategy that attempts to achieve the same objectives of our work over wired networks. However, GIA relies on random walks, which are not suitable for MANETs due to the much higher transmission error rates and dynamics of wireless networks. Furthermore, mobile P2P topologies have a topology that is more similar to the physical topology, and as such present hot spots at the center of the network. This requires an admission control scheme with back propagation, in order to identify hot spots that may be far from the sender, which is not the case of GIA. Our work achieves such back propagation by assessing the network conditions of a larger neighborhood.

III. Designing Load-Regulating Controllers

As mentioned in the introduction, the objective of the proposed controllers is to regulate the load on the network by adapting the configuration of the mobile P2P application. This regulation ensures that the load of the mobile P2P application is acceptable, so that the network does not become congested and as a consequence starts to drop packets and packet queues begin to build up. Congestion reduces the hit rate of the P2P application, since queries are lost due to collisions on the MAC layer and packet discards due to full queues. Further, congestion increases the response time, again due to the need for more retransmissions on the MAC layer and also due to queueing delays at the routing layer. The P2P application, however, wishes to send as many messages as possible, in order to answer queries faster and to increase the amount of nodes being queried (to increase the *coverage* of the query).

Thus, the controller should regulate the amount of messages sent by the mobile nodes running the P2P application, sending as much messages as possible, however avoiding the occurrence of collisions and large packet queues. The proposed controllers actuate on the TTL parameter of the P2P application, reducing it when the network load is high,and increasing it when the network load is low. Most unstructured and hybrid mobile P2P protocols employ parameters such as TTL that limit the depth of the search. As a consequence, the proposed controllers could be employed on those protocols without modifications. There are two challenges in building load-regulating controllers. First, one must assess the congestion of the network, which cannot be measured directly. Second, the controller must actuate over the P2P application in order to maintain an optimal load.

Two approaches were investigated for the autonomic controller. The first, based on the classic AIMD algorithm of TCP, uses the well known trade-offs of flow control in order to reduce the network load. The second is based on a machine learning algorithm and it infers the best control decision based on a set of training instances.

A. AIMD-Based Controller

The AIMD-based controller borrows the ideas of flow control employed in TCP [18]. It additively increases the TTL of the messages when the load is acceptable, however it decreases the TTL exponentially when a congestion is detected.

In order to identify a congestion, the controller employs the average length of the interface queue, as in previous works [16]. The length of the interface queue provides an estimate of network congestion, since longer queues indicate that the MAC layer is receiving more requests that it can handle. The length of the interface queue of the nodes in the vicinity is aggregated employing the equations below, where $0 \leq o \leq 1$ is the occupation of the packet queue. Thus, for $o = 1$ the queue is full, while $o = 0$ indicates and empty queue.

$$n_{i,k} = \begin{cases} 1 & \text{if } k = 0 \\ |V_i| & \text{if } k = 1 \\ \sum_{j \in V_i} n_{j,k-1} & \text{if } k > 1 \end{cases} \quad (1)$$

$$o_{i,k} = \frac{\sum_{j \in V_i} (o_{j,k-1} \times n_{j,k-1})}{n_{i,k}}, \quad k > 1 \quad (2)$$

The variable $o_{i,k}$ aggregates the occupation of nodes in the vicinity of i within k hops, while $n_{i,k}$ approximates the number of neighbors of the same node for k hops. The equations above perform an approximation of the mean, since the same node can be accounted for more than once. This calculation is also more practical than the precise mean. In order to calculate $o_{i,k}$ for a given k each node broadcasts k values to its peers. To calculate the precise mean, each peer would have to contact al the peers within the k hops neighbourhood. We aggregate the occupation on the vicinity because although a node may have a small occupation, its transmissions may be affected by nodes in the vicinity that have a high occupation. We employed a single input perceptron to identify if the current occupation characterizes a congested network. Following [19] definition, we created a perceptron with the input function described in Equation 3 and the logistic activation function, described in Equation 4.

$$E(o) = \beta \times (o - c) \quad (3)$$

$$G(x) = \frac{2}{1 + e^{-x}} - 1 \quad (4)$$

The behavior of the perceptron is described in the Equation 5. Positive output values indicate a congestion, while negative values indicate an uncongested network. Also, the absolute value of y indicate the certainty of the classification. The root of the F function (i.e, the congestion threshold) is c, whose determination is described in Section IV-A. Deviations from c are amplified by β. The "S-shape" of the activation function copes well with small variations in the ideal congestion threshold, avoiding binary classifications.

$$y \mapsto F(o) \begin{cases} -1 \leq y < 0 & \text{if the network is not congested} \\ 0 < y \leq 1 & \text{if the network is congested} \end{cases} \quad (5)$$

The second phase of the controller employs the classification to act on the system. Previous studies on flow control in the Internet indicate that additive traffic increase is more indicated when the network is not congested, while an exponential traffic decrease is best when the network is congested [18]. The new TTL in the following interation k of the algorithm will be given by $TTL_k = TTL_{k-1} + \Delta(o, k)$, where the difference $\Delta(o, k)$ is calculated as follows.

$$\Delta(o, k) = |F(o)| \times \begin{cases} \delta & \text{if } F(o) < 0 \\ (\alpha - 1) \times TTL_{k-1} & \text{if } F(o) > 0 \end{cases} \quad (6)$$

Equation 6 employs two parameters, $\alpha < 1$ and $\delta > 0$, used to control the amount of change on the multiplicative and addictive actions, respectively. Thus, the controller operation is similar to the congestion control mechanism of TCP, however the intensity of the adjustment is multiplied by the certainty of the classification ($|F|$), dealing better with uncertainty.

B. Machine Learning-Based Controller

The second autonomic controller was designed using machine learning techniques, and is based on the following observation: the highest network performance of any mobile network occurs when the network is near congestion. Under this regime, the network is forwarding as queries as possible, and the delay is still acceptable. A network under congestion is operating at its peak throughput, however the delay starts to increase exponentially due to packet collisions and longer packet queues [20]. Thus, the controller changes the TTL of the queries in order to achieve the peak throughput of the network, in a situation that is near congestion. The controller has three possible actions, as follows:

- `Keep` the TTL unaltered. This should occur when the network is already in the desired congestion state, achieving a throughput that is close to that of the saturation point, however the performance is acceptable.
- `Increase` the TTL. This action should be triggered when the controller believes that the network is not suffering from congestion, and an increased TTL would not lead the network into a congested state, where the performance would degrade due to collisions.
- `Decrease` the TTL. This action should be performed when the controller identifies that the performance is poor due to a significant amount of packet drops, long packet delays and other effects of excessive congestion.

In order to achieve those actions, the controller employs a machine learning (ML) classifier that predicts whether the network is congested or not. This classifier can be modeled as a function C that, given a set of inputs i, as well as the current TTL τ, outputs whether the network is congested or not, as described in Equation 7:

$$C(i_0, i_1, \ldots, i_n, \tau) \mapsto \{0, 1\} \quad (7)$$

The controller uses two instances of C to identify the next value of the TTL, as follows. Given the set of inputs at a given time ($i_{0,t}, i_{1,t}, \ldots, i_{n,t}$, written as I_t for short) and the instantaneous TTL τ_t, the controller calculates $C(I_t, \tau_t)$ and $C(I_t, \tau_t + 1)$. The four possible output combinations are shown in Table I, as well as the actions that taken on each of them.

The actions are intuitive: If the network is classified as not congested, and the classifier predicts that increasing the TTL will not make the network congested, the controller increases

the TTL. If the classifier believes that the network is not congested using the current TTL, however a higher TTL will most probably congest it, then the controller keeps the current TTL. Finally, if the controller identifies that the network is congested using the current TTL value, it decreases the TTL[1].

TABLE I
CONTROLLER ACTIONS BASED ON THE CLASSIFIER'S OUTPUT

$C(I_t, \tau_t + 1)$ \ $C(I_t, \tau_t)$	0	1
0	Increase	Decrease
1	Keep	Decrease

Each node on the network will run an instance of the controller. The input variables were defined as follows. All the input values can be measured locally or using the occupation estimation algorithm, mentioned in the previous section. Thus, it does not incur in any significant communication overhead.

- Local Occupation
- Neighbors' occupation
- Two hop Neighbors' occupation
- Current TTL

The controller employs a supervised learning algorithm to derive the hypothesis for its classifier. The training set was built using simulations, where the mean values of the input variables were measured after the network had reached its stationary state. Next, the network state was classified as congested or not based on the mean response time of the P2P queries. If the response time was higher than a certain threshold T, the network was classified as congested[2]. This threshold will depend on the characteristics of the network (described in details in Section IV-B). The controller actuates on the system on periodic intervals, which are pre-configured in the algorithm.

C. Supporting Real-Valued TTLs

In most mobile P2P implementations, the TTL is an integer value. However, we found out during the course of this work that the controllers could achieve a finer grained control when the P2P application supports real-valued TTLs. This requires a simple modification on the application, where the protocols could either employ floating points to represent the TTL, or approximate this behavior using a fixed point representation.

Real-Valued TTLs work as follows. Suppose that a peer issues a query with an initial $TTL = 3.5$, as depicted in Figure 1. This query is forwarded to the neighbors of the peer, which will decrement the TTL by one. Thus, the neighbors of the query source will forward the query with $TTL = 2.5$. The forwarding occurs normally until the TTL reaches a value $0 < TTL < 1$. In this case, the peer receiving the query will stochastically choose if it will forward the query, with a per-node forwarding probability equal to TTL. For example, when a node receives a query with $TTL = 0.5$, it will forward the query to each of its peers with a probability of 50%.

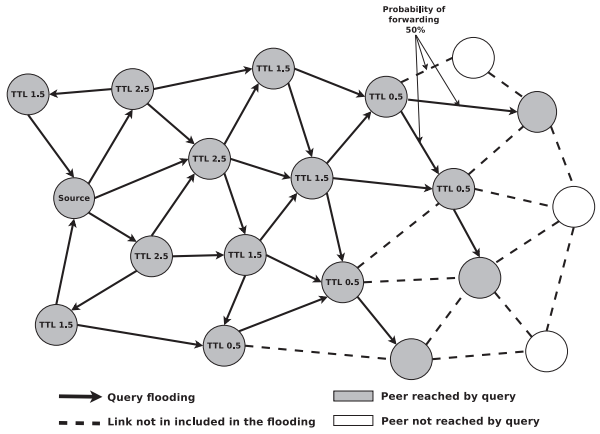

Fig. 1. Operation of real-valued TTLs.

IV. TUNING THE CONTROLLERS

Before evaluating the effectiveness of the controllers, each parameter was empirically tuned using simulations. The simulations employed the same environment of the performance evaluation, which is described in section V-A.

A. AIMD Controller

The first parameter to be analyzed was α, which controls the level of multiplicative decrease in TTL when a congestion occurs. This parameter influences how quickly it will react when congestion is detected. Smaller values of α will generate a more aggressive response, reducing the congestion in less time. This is seen in Figure 2, in which the response time grows with α, due to more collisions. Interestingly, the hit rate reduced by up to 4% with smaller α, as shown in Figure 3. This occurred because, although smaller values of α reduce the collision, they also reduce the coverage of the query, reducing the hit rate. Based on these results, $\alpha = 0.8$ was selected for the rest of the simulations since it presents a compromise of hit rate and response time.

Next, the value of δ was tuned. δ dictates the amount of additive increase in case of uncongested conditions. The best value of δ was 0.1, since it optimized both the hit rate (Figure 4) and the response time. The hit rate, for example, increased up to 5% for $\delta = 0.1$ with regards to $\delta = 0.3$. This shows that a more conservative approach in the increase yields less contention on the medium.

The last parameter is c, the congestion threshold. This value was obtained by observing the mean size of the queue when the performance starts to degrade, that is, when the hit rate begins to reduce and the response time increases exponentially. Table IV-A presents the mean occupation obtained when the performance started to degrade for different values of TTL. The second column indicates the amount of queries per second

[1]The state $C(I_t, \tau_t) = 1, C(I_t, \tau_t + 1) = 0$ should not happen in practice, however the default action was set to Decrease.

[2]Other parameters could be used for the classification, such as the average hit rate, average queue occupation, etc. We employed the response time since it is a performance metric that is perceptible by the user, and its value is correlated to the occurrence of collisions and queue buildups.

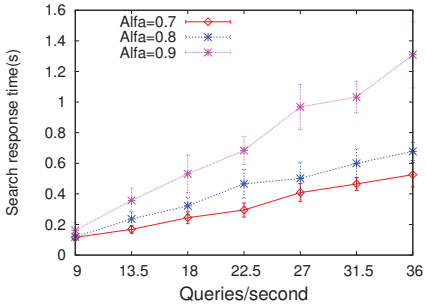

Fig. 2. Evaluating α – response time.

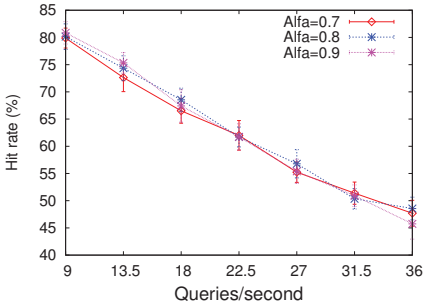

Fig. 3. Evaluating α – hit rate.

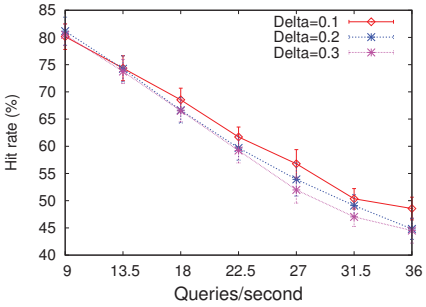

Fig. 4. Evaluating δ – hit rate.

TABLE II
AIMD CONTROLLER – DEFINING THE CONGESTION THRESHOLD.

TTL	Queries per second	Occupation
3	50	1.84%
4	25	1.05%
5	20	2.06%
6	15	1.67%
7	15	2.80%

that generated this degradation for different query loads. The threshold was defined as the mean value of the occupation plus one standard deviation, that is, $c = 0.027$.

B. Machine Learning Controller

The first tuning performed in the machine learning controller (MLC) was the selection of the classification algorithm. The selected algorithm was the one that presented the smallest amount of false negative and false positives using the training set. The training set contained tuples $\{i_{1,j}, i_{2,j}, ..., i_{n,j}, o_j\}$, composed of the n input variables and the output for each training instance j. Each training instance indicated the input variables for one node in one simulation. The output was set to zero if the network was not congested, that is, the average response time was inferior than T, and set to one otherwise.

The training set consisted of 1.2 million observations, which were generated from hundreds of runs of the network with the base setup described in Section V-A. In order to cover a large number of possible networks, we varied the number of peers, the load, the TTL used in the network. The TTL was fixed during the simulation.

For all the experimented threshold values, the algorithm that achieved the best amount of correct classifications in the Weka software [21] using cross-validation with 10 subsets was REPTree [22], with an accuracy of 87.41%. Next, the response time threshold was identified using simulations, in order to pick the best value based on network metrics such as delivery rate and energy consumption. The simulations varied the amount of queries per second, in order to evaluate the adaptability of the controller.

Figures 5 and 6 present the response time and the hit rate, respectively, for different values of T. The first figure shows that the controller achieves its goals of providing an upper bound on the response time, since no curve crosses the threshold T. This behavior, however, is obtained at the cost of a decrease in the hit rate for higher values of queries per second, since the peers must reduce their TTL in order to reduce the load on the network. As a consequence, the queries reach less nodes, reducing the hit rate. The remainder of the paper uses $T = 2s$, since it showed a compromise of hit rate and response time.

Next, δ was empirically adjusted, again looking at the response time and hit rate. Values from 0.25 up to 1.0 were evaluated, and $\delta = 0.25$ was chosen because this value maximized the hit rate (Figure 7), while presenting a good response time, as shown in Figure 8.

V. PERFORMANCE EVALUATION

A. Simulation Setup

The simulated scenario, instantiated on NS-2, approximates the workload of a MANET employed for intelligent traffic applications in an urban environment. The protocols run over UDP, since TCP does not perform well on wireless links. The routing layer employs OLSR, because it is the most employed protocol in real deployments, and IEEE 802.11 in the MAC/PHY layers. The simulations model a Cisco Aironet 350 radio [23], having a transmission output of 10 dBm. This radio consumes 1.6887W, 1.0791W and 0.6699W while in the transmit, receive and idle modes, respectively. Packet losses due to interference or signal variation were not considered. The routing layer had a packet queue of 30 packets.

Fifty nodes are deployed in an area of 1200x1200m, following an uniform distribution. Nodes move using the restricted random waypoint (RRWP) model, where nodes only follow certain pre-defined paths that model a section of Houston [24]. The chosen RRWP implementation ensures the existence of a stationary distribution of node speeds [25]. Nodes move from one point to another within the streets of Houston following the speed limits of each section of the street. Upon arriving at their destination, they pause for some time and select another destination. The mean pause time was 10s and its standard deviation was set to 2s.

Besides, 50% of the nodes are always on the network, while the remaining nodes enter or leave the P2P network (following an uniform distribution) from time to time, to simulate churn. The P2P network has 250 different files in total, each of them

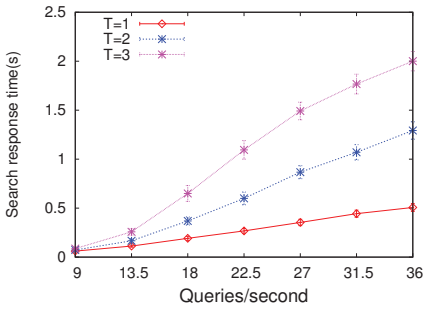

Fig. 5. MLC, congestion threshold - response time.

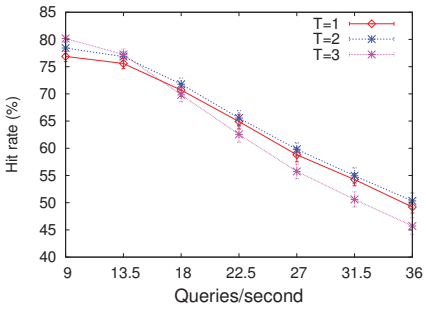

Fig. 6. MLC, congestion threshold - hit rate.

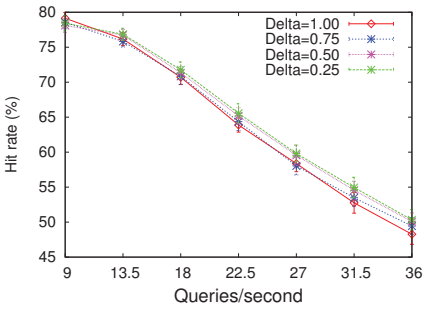

Fig. 7. MLC, δ - hit rate.

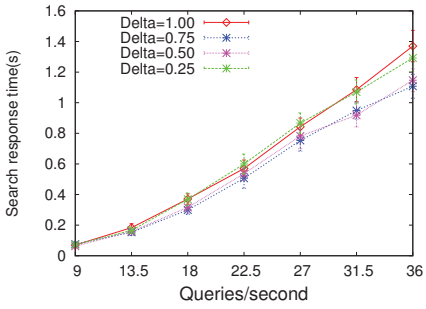

Fig. 8. MLC, varying δ - response time.

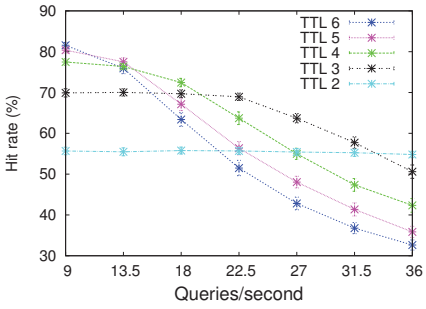

Fig. 9. Static TTL configuration – hit rate.

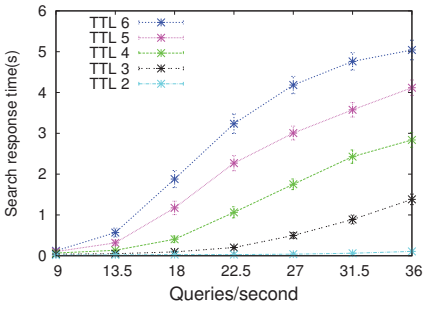

Fig. 10. Static TTL configuration – response time.

having five replicas. The total simulation time is 550s, enough time to reach a stationary behavior, and file queries occur only after 50s in order to allow the nodes to build routes.

Each peer runs an instance of Gnutella, with the number of neighbors set to three. Although this paper employed Gnutella, the controllers are applicable to any unstructured or hybrid mobile P2P protocol, as discussed in Section III. Each scenario was simulated 30 times with different random seeds, and the results present their mean with a 95% confidence interval.

B. Best Manual Configuration

This section evaluates the performance of static configurations, which are later used to estimate the *best manual configuration*. In the following sections, the *best manual configuration* (BMC) is the best *fixed* TTL value for each configuration of the network. Note that, since the TTL is fixed, it may not be the *optimal* or most performing configuration, since the network could profit from heterogeneous configurations for each query to optimize its performance. However, the BMC is useful to measure *how good* a certain local adaptation strategy is when compared against an omniscient entity choosing a fixed TTL.

Figure 9 shows the hit rate and that no single TTL value outperforms the others. Instead, the best TTL value will depend on the amount of queries per second. Higher loads will favor smaller TTLs, since they will generate less messages on the network, and thus less collisions and delays. Meanwhile, lower query rates will tend to benefit from higher TTLs, because the queries will reach more peers.

The response time, on the other hand, presents a different behavior. Figure 10 shows that smaller TTLs have lower response times. The curves present an almost exponential increase, and when the network is congested the response time starts to increase linearly. This is due to three factors. First, since the queries with smaller TTLs will reach less distant peers, the peers responding to a query will be closer to the source of the query. Second, limiting the reach of the query reduces the amount of messages on the network, reducing contention. Thus, the choice of the best TTL is a compromise among the hit rate and the response time. The linear behavior of the response time when the network is congested is due to collisions and packet queues. When the amount of collisions reaches a threshold, the back-off mechanism of IEEE 802.11 stabilizes at the maximum contention window. Similarly, packet queues discard extra packets when they are full, limiting the forwarding rate.

C. Comparison Against the State of the Art

This section shows a comparison of the proposed controllers against the state of the art of self-configuring P2P networks as well as the best manual configuration.

The proposed solutions are compared against Expanding Rings (ER) [26], which is an extension of Gnutella for wired networks that attempts to reduce the load induced by the P2P network. The authors propose an adaptive TTL that works as follows. The query starts with an initial TTL t_0, which is fixed. If no results are found for the query after a certain time interval, the query is retransmitted, this time with a higher TTL $t_1 = t_0 + \delta$, where δ is a fixed increment. This process is repeated either until the query returns a hit, or when the TTL reaches a maximum value. ER has the drawback of a

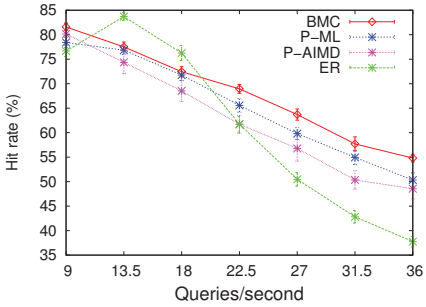

Fig. 11. SoA comparison – hit rate.

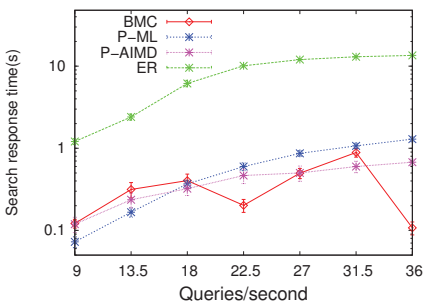

Fig. 12. SoA comparison – response time.

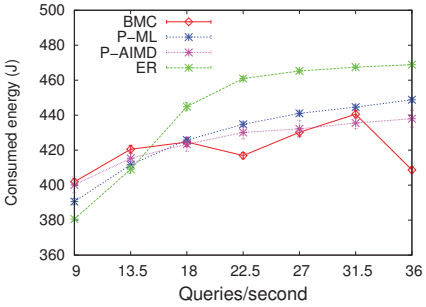

Fig. 13. SoA comparison – consumed energy.

high response time for queries searching for unpopular items. However, it is very efficient for popular items, since they should be replicated in several nodes. We evaluated differents ER parameters in order to obtain performance and adaptability. The best set of parameters were used in our comparisons, namely $t_0 = 2$, $\delta = 1$ and Maximum TTL = 7.

Besides expanding rings, the results present the best manual configuration for each query rate in Section V-B, which is depicted in the curves labeled "BMC". The best manual configuration is the one with the highest hit rate. Because each point may use a different TTL value, the "shape" of the curve can be very irregular. Again, this curve is not the optimum value, however it is included as a reference to measure how good the adaptive methods perform against an Oracle which chooses a *fixed TTL for the entire simulation*, based on omniscient knowledge, that maximizes the hit rate. The optimum curve, which will be simulated in future work, would be obtained from an oracle that picks *the best TTL for each query* independently.

Figure 11 presents the hit rate. As expected, the BMC curve obtained better results when compared to the controllers due to its omniscience. However, both the AIMD and ML presented results closer to the BMC curve. BMC outperformed P-ML by 2-5%, while it outperformed P-AIMD by 2-7%. Expanding rings, performs better than the controllers for small loads, since it employs a very conservative sending strategy. However, it performs poorly for loads over 22.5 queries per second, because its mechanism cannot distinguish failed queries due to collisions from failed queries due to small TTLs. As a consequence, ER tends to increase the TTL even more when the network is congested.

The main drawback of ER is the response time, as shown in Figure 12. Since ER does not learn from past queries, each new query starts with a small TTL, generating several iterations of the algorithm. The controllers, on the other hand, presented a much smaller response time, which grew with the amount of queries per second. Still, the response time did not increase exponentially as in the manual TTL configurations shown in Section V-B. This behavior is due to the fact that the controllers decrease the TTL with higher loads. The response time of the controllers is comparable or lower to the response time of the best manual configuration. The controllers achieved lower response times than the BMC because they employ variable TTLs, while the BMC is limited to a fixed TTL.

The mean energy consumption per node is shown in Figure 13. It resembles the curve of the response time, since both are a function of the amount of messages sent. However, the difference in energy consumption is less pronounced, since it accounts for the consumption of the radio during the idle and reception states. The controllers consume slightly more energy than the BMC, since the BMC was devised based on global knowledge. On the other hand, the controllers consume less than ER, due to less collisions and retransmissions.

VI. Conclusions and Future Work

Configuring an application or a network to achieve optimum performance is a hard task, since the choice of parameters will depend on the load and topology of the networks. This problem is even harder on wireless networks, since the topology usually changes with time. Mobile P2P networks also suffer from this problem, namely due to network congestion, which may lead to an increase in response times and less queries being correctly resolved.

In order to address those issues, this paper proposed two closed loop controllers that dynamically manage the configuration of the P2P protocol, adapting it to the current network conditions. One of the controllers employs the flow control strategy employed in TCP, while the second controller uses machine learning techniques. Both controllers assess the existence of congestion by measuring localized information and actuating on the TTL of the queries. Thus, the proposed solutions are ideal for ad hoc and mesh networks, because they do not require a central decision point.

The simulations compared the proposed controllers against an existing adaptive protocol, as well as an omniscient oracle that chooses the best fixed TTL configuration for each network. The controllers achieved a performance that is comparable to that of the oracle in terms of energy consumption and response time, and presented a hit rate up to 7% lower.

As future work, we intend to tune other parameters of the P2P application at the same time, for example the number of neighbors and their TTL. We will employ unsupervised machine learning techniques to make the controllers more robust. We will implement the controller on other types of P2P applications, for example hierarchic or hybrid approaches.

ACKNOWLEDGEMENTS

The authors would like to thank CNPq, CAPES and FAPEMIG, brazilian research agencies, for their financial support in this work.

REFERENCES

[1] M. Papadopouli and H. Schulzrinne, *Peer-to-Peer Computing for Mobile Networks*, 1st ed. Springer, 2009.

[2] X. Shen, H. Yu, J. Buford, and M. Akon, *Handbook of Peer-to-Peer Networking*, 1st ed. Springer, 2009.

[3] D. N. da Hora, D. F. Macedo, L. B. Oliveira, I. G. Siqueira, A. A. F. Loureiro, J. M. Nogueira, and G. Pujolle, "Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks," *Elsevier Computer Communications*, vol. 32, pp. 1445–1459, July 2009.

[4] L. B. Oliveira, I. G. Siqueira, and A. A. F. Loureiro, "On the performance of ad hoc routing protocols under a peer-to-peer application," *Journal of Parallel and Distributed Computing (JPDC)*, vol. 65, no. 11, pp. 1337–1347, November 2005.

[5] F. P. Franciscani, M. A. Vasconcelos, R. P. Couto, and A. A. F. Loureiro, "Peer-to-peer over ad-hoc networks: (re)configuration algorithms," *Journal of Parallel and Distributed Computing (JPDC)*, vol. 65, no. 2, pp. 234–245, February 2005.

[6] M. Conti, E. Gregori, and G. Turi, "A cross-layer optimization of gnutella for mobile ad hoc networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2005, pp. 343–354.

[7] R. Zhang and Y. C. Hu, "Assisted peer-to-peer search with partial indexing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1146–1158, 2007.

[8] Z. Zhuang, Y. Liu, L. Xiao, and L. M. Ni, "Hybrid periodical flooding in unstructured peer-to-peer networks," in *International Conference on Parallel Processing*, 2003.

[9] H. Jiang and S. Jin, "Exploiting dynamic querying like flooding techniques in unstructured peer-to-peer networks," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2005, pp. 122–131.

[10] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 363–374, August 2008.

[11] H. Chen, H. Jin, Y. Liu, and L. M. Ni, "Difficulty-aware hybrid search in peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 1, pp. 71 –82, jan. 2009.

[12] N. Shah and D. Qian, "Context-aware routing for peer-to-peer network on manets," in *IEEE International Conference on Networking, Architecture, and Storage (NAS)*, july 2009, pp. 135 –139.

[13] V. Ramasubramanian, T. L. Rodeheffer, D. B. Terry, M. Walraed-Sullivan, T. Wobber, C. C. Marshall, and A. Vahdat, "Cimbiosys: a platform for content-based partial replication," in *Proceedings of the USENIX symposium on Networked systems design and implementation (NSDI)*, 2009, pp. 261–276.

[14] T. Hara and S. K. Madria, "Consistency management strategies for data replication in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 7, pp. 950 –967, july 2009.

[15] D. F. Macedo, A. L. dos Santos, J. M. S. Nogueira, and G. Pujolle, "A Knowledge Plane for Autonomic Context-Aware Wireless Mobile Ad Hoc Networks," in *IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)*, 2008, pp. 1–13.

[16] D. F. Macedo, A. L. dos Santos, J. M. Nogueira, and G. Pujolle, "Fuzzy-based load self-configuration in mobile P2P services," *Elsevier Computer Networks*, vol. 55, no. 8, pp. 1834–1848, 2011.

[17] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," in *SIGCOMM*, 2003, pp. 407–418.

[18] D.-M. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN*, vol. 17, no. 1, pp. 1–14, 1989.

[19] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Prentice hall, 2010.

[20] J. He and H. K. Pung, "Performance modelling and evaluation of ieee 802.11 distributed coordination function in multihop wireless networks," *Computer Communications*, vol. 29, no. 9, pp. 1300–1308, 2006.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[22] J. R. Quinlan, "Learning with continuous classes," in *5th Australian Joint Conference on Artificial Intelligence*, 1992, pp. 343–348.

[23] Cisco Systems, "Cisco Aironet 802.11abg cardbus adapter," http://www.cisco.com/en/US/prod/collateral/wireless/ps6442/ps4555/ps5818/product_data_sheet09186a00801ebc29.html, August 2012.

[24] A. K. Saha and D. B. Johnson, "Modeling mobility for vehicular ad-hoc networks," in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks (VANET)*, 2004, pp. 91–92.

[25] J.-Y. le Boudec and M. Vojnovic, "The random trip model: stability, stationary regime, and perfect simulation," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1153–1166, 2006.

[26] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search for peer-to-peer networks," in *Third International Conference on Peer-to-Peer Computing (P2P)*, Sep. 2003, pp. 102–109.