Addressing Non-Homogeneities in a Ubiquitous P2P Platform for Context Exchange

Ayis Ziotopoulos (ayis.ziotopoulos@utexas.edu), Gustavo de Veciana (gustavo@ece.utexas.edu)

Abstract

We propose mechanisms to combat the congestion caused by non-homogeneities in the distribution of information, the peers' locations, and the demand for information in a peer-to-peer system for exchanging context. We propose two novel mechanisms for adapting the topology formed by the peers to the underlying traffic: our first mechanism employs virtual locations for the peers while the second modifies the edges connecting them. The positive effect of our mechanisms is evaluated experimentally. Our key metric is the average query delay, a critical performance indicator in a prototypical system we have already presented, expected to be used by mobile users for exchanging contextual information.

I. Introduction.

Recent developments in hardware design and manufacturing have made possible the advent of devices with capabilities in computing, sensing, and, mobility that reach, if not surpass, the expectations for ubiquitous computing laid out by M. Weiser in the publication that introduced his vision to the world [1]. Still, there exists a gap between what the state-of-the-art in computing fabric can offer and the performance of ubiquitous systems.

Context-awareness has long been recognized by researchers, as a pre-requisite for ubiquitous computing. Informally, context-awareness refers to the ability of an application to recognize the 'environment' in which it executes. For example, a mobile user equipped with a sensor-enabled smart-phone, opportunistically gathering/contributing information relevant to the available applications in his vicinity is a prime use-case of context-awareness in ubiquitous computing.

We envisage a system where computing devices, e.g., smart-phones, robots, and, associated applications will contribute and consume opportunistically contextual information relevant to a 'region' in space/time around them. In our vision, we cater explicitly for the exchange and storage of contextual information as mobility and the opportunistic nature of the applications interested in it pose several challenges, e.g., for the persistent 'floating' of information in its relevant space [2]. We articulated a concrete implementation of our ideas in [3]. Our platform consists of a peer-to-peer (p2p) overlay where contextual information relevant to a region is exchanged by mobile users via peers that are 'responsible' for storing the context pertinent to the area around them and answering queries about it.

Despite our design decisions to enhance our platform's performance, e.g., information to be stored close to its expected consumption area, we propose to add overlay edges between peers that act as 'shortcuts' for traffic, the challenges posed by an inhomogeneous operating environment can significantly hamper its performance. For example, consider a ubiquitous application where users opportunistically provide and query real-time information about the waiting time in participating restaurants. The spatio-temporal distribution of the information input in such an application is not uniform, e.g., during weekends information about locations around the downtown of a city may be more likely to be present in the system. Peers located close to downtown may then be disproportionally congested.

Contributions. In this paper, we build on our previous work and present mechanisms to tackle the load-balancing challenges posed by the presence of non-homogeneities in ubiquitous platforms. More specifically, we address issues relevant to the spatio-temporal distribution of the information to be exchanged/stored in our system, the distribution of the peers available to store that information, as well as the distribution of the queries to acquire that information.

We propose two novel mechanisms: employing *virtual* locations for the peers adapted to the underlying traffic as well as adapting the *edges* among them. The first mechanism is inspired by our intuition that traffic at a peer consists of two components: *local* traffic, due mainly to the relevant location of a peer to its neighbors, and *end-to-end* traffic, due mainly to the location of a peer, the more 'central' the location of a peer, the more routing traffic the peer will suffer. We propose a novel distributed load-balancing algorithm based on estimations of queue sizes of neighboring peers. Our algorithm increases the density

978-1-4673-1239-4/12/$31.00

of peers in areas with high traffic in order to balance the overall load among the participating peers.

The second mechanism, i.e., adapting the overlay edges to the underlying traffic aims to create traffic-aware edges that 'match' peers with high production of queries to peers with high 'consumption'. We propose a novel distributed algorithm that estimates traffic between source-destination pairs of peers and creates edges between pairs with maximal traffic. An experimental evaluation of our proposed mechanisms via custom simulations reveals their beneficial effect on average query delay.

Related work. Load-balancing in P2P networks is a problem of recognized importance, see, [4], [5], and, [6], where the objective, as in our work, is to alleviate the effects of the non-uniform storage load per peer and the non-uniform distribution of queries. All the works referenced live in the realm of distributed hash table (DHT) p2p systems. In contrast, in our work data are stored based on their location instead of an id derived from the content. As a result, we virtualize peers' locations to balance the load instead of, e.g., creating virtual servers to hold parts of the identifier space as in [4]. Additionally, we create edges that work as 'optimal' shortcuts for the query traffic as opposed to, e.g, [6], where edges are created to connect peers storing adjacent parts of the identifier space. Our solutions are fully distributed involving neighbor peers only, unlike the approach in [4] where load information is published in a centralized directory. Our approach, creates minimal overhead to the network as traffic propagation is limited in the vicinity of neighboring peers.

Organization. §II briefly reviews the main elements of our platform introduced in [3], i.e., how peers store information, how peers are connected to each other and the *range* query which is our fundamental building block for more complex queries. In §III we describe the two mechanisms we offer to combat the effect of non-homogeneities and discuss some first experimental results showing the benefit derived. Finally, §IV concludes with a discussion of ongoing and future work.

II. Platform architecture.

In our platform nodes can contribute events, make queries, and/or serve as a peer in the p2p overlay network. We assume that an infrastructure of wired fixed devices, e.g., PCs and corporate servers, serve as peers and act as proxies for mobile wireless nodes e.g., sensors, phones, etc., which can not serve as peers. All entities are assumed to know their locations, but exact locations are not necessary. Fig. 1 exhibits the elements of the platform which will be discussed below.

A. Event model and storage.

Our focus is on capturing a spatio-temporal flow of (short-lived) events which can represent a wide range

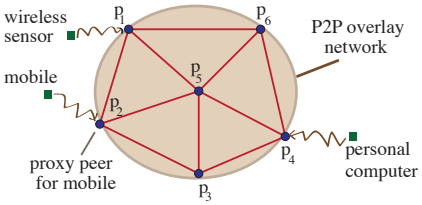

Fig. 1. Elements of the p2p architecture.

of data/information associated with users, applications, sensors or machines. An informal view of our event model can be visualized as a cylinder, see Fig. 2, where an event e has associated spatial coordinates, *e.location* indicating where the event 'occurred' or is centered, a range *e.range* defining the region where it is relevant, as well as a time at which it is generated/starts and duration: *e.time* and *e.duration* respectively. For simplicity, we denote the disc centered at location $e.location$ with radius $e.range$ as $B(e.location, e.range)$ and refer to events whose duration contains the current time as *active*.

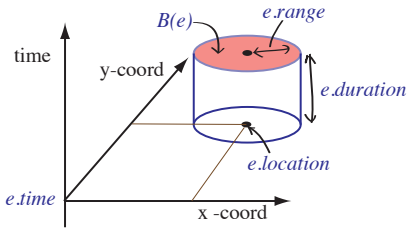

Fig. 2. Event model.

Opportunistic gathering/contributing of information is more likely to occur for events in the vicinity of the consumer/reporter. The previous 'locality' observation motivates the following rule:

Rule 1. Event storage and deletion. *An event e is stored at the overlay peer p which is closest to $e.location$.*

B. Overlay topology management & routing.

We let $\Pi = \{p_1, p_2, \ldots\}$ denote the set of peers identified by their unique *virtual* locations $p_i \in \mathbb{R}^2$. Initially, a peer's virtual location equals its physical location. In the sequel, whenever we talk about a peer's location we mean its virtual location. They are interconnected via links in a structured p2p overlay, represented in Fig. 1 by thick straight lines. The basic overlay connectivity is driven by the spatial locality of queries – it corresponds to the Delaunay graph induced by the peers' locations. See [7] for the definition of the *Voronoi tessellation* $V(\Pi)$ induced by $\Pi \subset \mathbb{R}^2$, the cell $C(p|\Pi)$ corresponding to a peer p and the *Delaunay Graph* (DG) induced by $\Pi \subset \mathbb{R}^2$.

Assumption 1. ***Topology and Routing.*** *We assume the overlay connectivity of our platform is a* superset *of the*

Delaunay graph. We further assume peers employ greedy routing to store events and make queries over the overlay.

Greedy routing here, refers to a policy where each peer forwards a message (event or query) destined to a location, say l, to its closest neighbor to l with the intent of eventually reaching the peer closest to l.

Greedy routing on a superset of the DG always succeeds thus, we will consider additional overlay links to reduce the number of hops a query has to traverse. In the sequel we will use the well known result of [8] which we adopt as follows: a peer need only generate a location at random according to a distribution proportional to the inverse of the square of the distance from its own location, and then route a message to that location, i.e., the closest peer to the randomly selected location. In the sequel we denote such additional edges, 'Kleinberg' edges.

C. Query model and processing.

The goal of the infrastructure is to efficiently support spatio-temporal queries. In this paper, we focus on *range queries* on *active* events. Range queries are defined as follows:

Definition 1. *The* **Range Query** *$RQ(l, r)$ returns all active events e within a range r of a location l, i.e., events currently stored in the system such that $e.location \in B(l, r)$.*

For example, return all users/resources currently available within 200m of my location.

A detailed algorithm for resolving range queries is described in [3]. For simplicity, we will focus on 'point' queries, i.e., queries with zero range around a peer's location. These queries return all the events stored in that peer. As far as performance is considered, these queries consist of the exchange of two messages between the source and destination peer, greedily routed on the p2p overlay.

III. Congestion load-balancing.

In this section, we explore ways to adapt the overlay topology, i.e., the locations of the peers and the edges connecting them, so as to balance the traffic load on the peers. As discussed in the introduction, different peers may receive widely different amounts of traffic due to non-uniformities in the peers' and/or traffic's spatial distribution.

'Local' vs. 'End-to-End' Congestion. Congestion at the peers can be conceptually divided into 'local' and 'end-to-end' congestion. We will define the former as caused by exceedingly high event traffic generated in a cell or query traffic whose source and destination lie in the same cell. The latter is caused by routing of queries across the overlay network. Local congestion depends on the size of a peer's cell; the bigger the cell the more local traffic it might be expected to receive. End-to-end congestion depends mainly on the location of a peer, the more 'central' a peer is, the more routes cross its cell. End-to-end congestion also depends on the quality of the edges in the topology; long edges connecting peers that share a lot of traffic are better.

In this section we consider the following two mechanisms to mitigate congestion: employing virtual locations for the peers and, adapting the connections among peers to the underlying traffic.

The first approach addresses mainly the local congestion and, the second approach addresses the end-to-end congestion.

A. Virtualising the peers' locations.

The locations of the peers uniquely determine the size of the cell of each peer. Consider the set P containing all the peer locations $\{p_1, \ldots, p_n\}$. We denote by $Q_i(P)$ the queue size of the i^{th} peer under the peer placement given by P. Ideally one would like to find an algorithm that provides a solution to the following optimization problem.

$$\min_{P}\{\mathbb{E}[\sum_{i=1}^{|P|} Q_i(P)]| \text{ for all } p_i \in P\} \quad (1)$$

under the constraint of greedy routing and a fixed average intensity of events γ_e and queries γ_q. By Little's law, this is a proxy for the average traffic delay in the system.

For local traffic *only*, dividing the region in, e.g., square grid, etc would be a solution for Eq. 1. The addition of 'end-to-end' traffic complicates things; the bigger the area of a peer's cell, the more traffic it receives. If a peer could advertise a different location than its current one, a more 'balanced' overlay topology could be achieved.

This observation motivates the following simple heuristic:

Rule 2. Move Heuristic. *An overloaded peer, p, may invite a non-overloaded peer, q, to drop its current location and join the overlay with a different 'virtual' location that is closer to p and thus take some of its load.*

For purposes of implementing this heuristic, a peer will be considered overloaded with respect to another peer if its queue size exceeds a fixed multiple, $f_{qs} > 1$, of the other peer's queue size. [1]

For simplicity, we will require the peers q and p involved in such move heuristics to be neighbors in the DG. The peer q will be restricted to move to a new location that lies along the line connecting its original location to the location of p. The resulting distance between the two peers will be a fraction, $f_d < 1$, of the original distance between them, see Fig. 3.

The above mentioned heuristic has the effect of reducing the area of the overloaded peer's cell, which in turn

[1] Our protocol for peers to join/leave the overlay as well as change their locations is omitted due to space limitations.

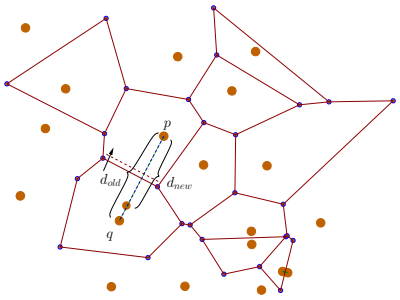

Fig. 3. Move Heuristic, d_{old} is the distance between p and q before the heuristic, d_{new} is the distance after and $\frac{d_{new}}{d_{old}} = f_d$.

will lower the local traffic the peer sees. Moreover, a peer with a big cell is likely to lie in the path of an increased number of queries and suffer increased end-to-end traffic as well. Thus, reducing a cell's size is likely to also reduce the end-to-end traffic it sees.

Evaluation. To evaluate the effectiveness of the 'move' heuristic on the mean delay to process an event, we performed the following experiment: we varied the average intensity of events arriving at a peer per m^2-sec, γ_e, and measured the mean delay to process an event with and without the 'move' heuristic, see Fig. 4. We assumed that the event arrival process is Poisson with mean rate γ_e, ranging from 0.2 to 2 events per m^2-sec. New event arrivals are assumed to be homogeneous in space, and arriving events enter the queue of the closest peer. To decouple the study of the local congestion from the end-to-end congestion we set $\gamma_q = 0$ for this experiment. The time to process an individual event is assumed to be an independent exponential random variable with mean $\mu = 1$. The overlay topology is generated by placing 60 peers independently in a 5×5 region according to a homogeneous Poisson process with rate $\lambda = \frac{60}{5^2} = 2.4$ peers per m^2. Two peers are connected if and only if they are neighbors in the Delaunay graph. For this experiment, since $\gamma_q = 0$, the Kleinberg edges would not play any role. For our preliminary evaluation of the 'move' heuristic, we used the following parameters $f_{qs} = 1.3$ and $f_d = 0.95$. Peers were selected to perform the heuristic at random, i.e, uniformly, at each tick of an exponential clock with rate $\mu_{move} = 0.05$. The rate of the clock has been selected such that on average each peer will have at least 100 event arrivals before being selected to implement the heuristic. The simulation lasted for 120000 units of time, ensuring that on average 100 'cycles' where performed, each cycle corresponding to a period in which every peer performs the heuristic at least once.

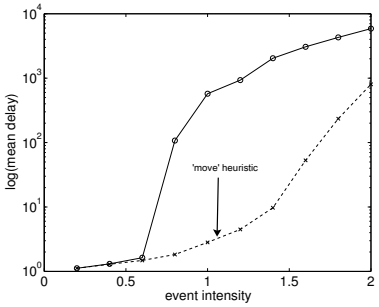

Fig. 4. Figure shows in logarithmic scale the mean delay to process an event as the spatial intensity of events γ_e grows in the overlay network. Two cases are shown, the first where peers are randomly located in space, and the second after the move heuristic is carried out.

The 'move' heuristic has an obvious beneficial effect on the mean delay to process an event for the entire range of event traffic. Although the maximum mean event arrival intensity at a cell is $\overline{\gamma} = \frac{\gamma_e}{\lambda} = \frac{2}{2.4} = 0.84 < \mu = 1$, due to statistical variations of the cell sizes, some peers will receive more traffic than they can handle and will become unstable. In Fig. 5 we show that the 'move' heuristic manages to create a homogeneous topology where most cells are of roughly equal size, thus minimizing the probability a peer overflows. This is consistent with our observation about Eq. 1.

Beneficial effect on 'end-to-end' congestion. To evaluate the effect of the 'move' heuristic on the mean delay to process a query, we performed the following experiment: we varied the intensity of query traffic generated per m^2-sec, γ_q, and measured the mean delay for a query starting from its source to reach its destination, with and without the 'move' heuristic. Throughout the simulation, each peer has a Kleinberg edge to a fixed point, the 'target' of that edge is the peer that happens to be closest to that point at any time. The initial placement of the peers, depicted in Fig. 5, is the same as the one used in our previous experiment. Due to space limitations, we omit the rest of the experiment details. The results are shown in Fig. 6. The 'move' heuristic appears to be beneficial for high query intensities; we further discuss that point in the sequel.

A rough, yet enlightening, view of the effect of our heuristic is that it divides the overlay topology in two areas: the inner area, which consists of uniformly arranged peers with small cells that suffer primarily from end-to-end congestion, and the outer area, which consists of uniformly arranged peers with big cells that suffer primarily from local congestion, i.e., with high probability queries originate

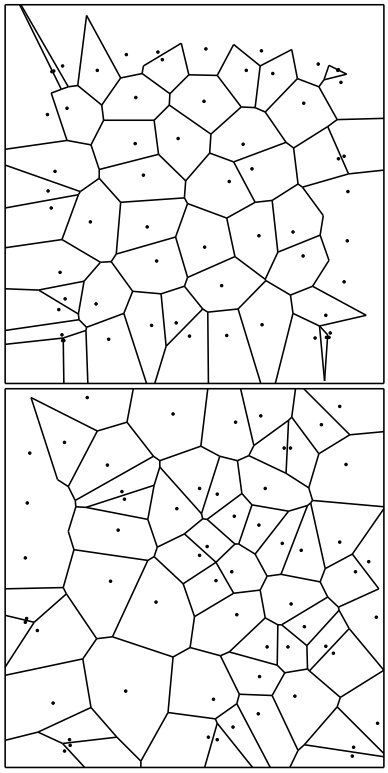

Fig. 5. Resulting overlay by applying the 'move' heuristic for $\gamma_e = 2$ (up) and original overlay topology (down) (the facets extending to infinity have been omitted).

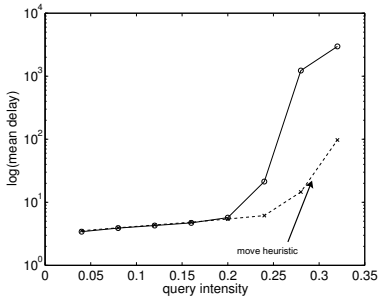

Fig. 6. log(mean delay to process a query) vs. the spatial intensity of queries γ_q.

and end at their cell. Fig. 7 exhibits the resulting topology achieved by our heuristic when $\gamma_q = 0.32$. The inner area is defined by the dashed disk. The resulting arrangement ensures that *all* peers receive on average the same amount of total traffic, local plus end-to-end.

B. Adapting the edges among peers to non-uniform traffic.

In Section II we proposed augmenting our overlay topology with Kleinberg edges. The effect of these edges is to create short, on average, routes between any two peers

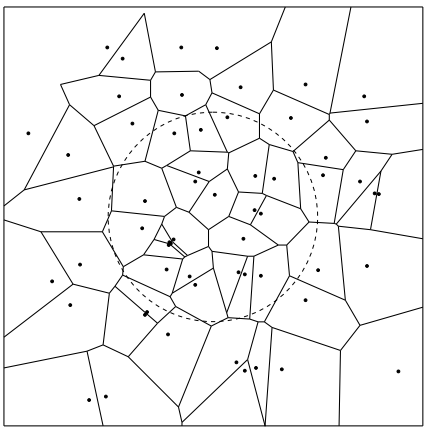

Fig. 7. Resulting overlay by applying the 'move' heuristic for $\gamma_q = 0.32$. The facets extending to infinity have been omitted.

in the overlay. A limitation of Kleinberg edges is that they are oblivious to the congestion in the network, and they are not sufficiently flexible to handle non-uniformities in the traffic. For example, consider the case of a group of people attending a conference at a convention center. Some of their queries could be destined to locations corresponding to the restaurant where the conference banquet will take place. In this case traffic is not homogeneous in space. The peers that happen to be in the path between them and the destination peer answering their queries will be disproportionally stressed by traffic.

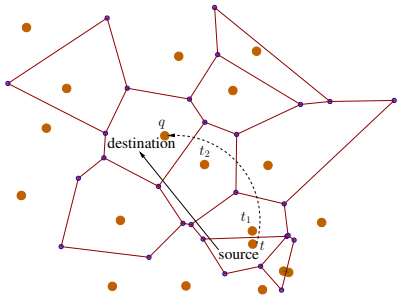

Fig. 8. A 'dipole' model for non-uniform traffic.

To model the non-uniformity in traffic, we introduce the concept of a traffic 'dipole', see Fig. 8. Each dipole creates traffic that originates from one location, i.e., its *source*, to another location, i.e., its *destination*. The distance between the source and the destination of a dipole is termed its *separation*.

To realize a traffic-aware p2p topology, one can further augment the topology by a traffic-dependent edge per peer. A simple heuristic would be:

Rule 3. Congestion Edge Heuristic. *Each peer, q, estimates the average traffic intensity, $\gamma_{t,q}$ and the average traffic delay, $d_{t,q}$, for the queries originating from every peer $t \in P$ in the network destined to q. The peer with the maximum $\gamma_{t,q} \times d_{t,q}$ is selected to create an edge to q. We call the associated edge, the* congestion *edge of peer t.*

The quantity $\gamma_{t,q} \times d_{t,q}$, by Little's law, is a proxy for the average number of queries 'in-flight' from peer t to peer q. By creating an edge between peers t and q, we offload the corresponding traffic from the overlay network and restrict it to the source/destination. In our implementation, each peer, t, maintains the cost, $\gamma_{t,q} \times d_{t,q}$ associated with the peer q to which it directs its congestion edge. In case another peer, r, invites t to establish its congestion edge toward it, t accepts only if $\gamma_{t,r} \times d_{t,r}$ exceeds $\gamma_{t,q} \times d_{t,q}$. Due to space limitations we omit the procedure through which a peer estimates the load and the delay from another peer.

Evaluation. We studied the performance of the congestion edge heuristic via the following experiment. We used the peer placement depicted in Fig. 5, the same as the one used in the experiments for the move heuristic. Additionally, we augmented the topology with edges derived from the Delaunay Graph as well as with Kleinberg edges. We simulated a combination of homogeneous and non-homogeneous traffic. The homogeneous traffic consisted of events arriving with intensity $\gamma_e = 0.05$ events/m^2-sec and queries arriving with intensity $\gamma_q = 0.04$ queries/m^2-sec. For the non-homogeneous traffic, we created a fixed number, 10, of traffic 'dipoles'. The source and destination of each dipole were placed at random on the topology. For simplicity, in our implementation all dipoles had a traffic intensity equal to $\gamma_{\text{non-uniform}} = 0.1$ queries/sec. The service time for each event/query was an independent exponential random variable with rate $\mu = 1$. The rate of the exponential clock triggering the heuristic at the peers was set to $\mu_{\text{congestion edge}} = 0.2$ and the simulation lasted for $T_s = 50000$ time units. The simulation duration was chosen so as to ensure that on average at least 100 cycles of the heuristic are performed, where a cycle is defined as the period where all the peers perform the heuristic.

In Fig. 9 we vary the dipoles' separation and plot the average delay for traffic. The traffic consists of events, uniform queries, and non-uniform queries. The intuition is that as the separation of the dipoles increases more peers will suffer from the resulting transit traffic.

The relative performance of the two schemes matches our intuition. Indeed, the mean delay achieved by our heuristic remains essentially the same regardless of the dipole separation, as opposed to the delay of the base case without our heuristic.

One could argue that the comparison we presented in our previous experiments is not fair since peers under our heuristic have one more edge to route queries, the congestion edge. Note that even if additional Kleinberg edges were added to the scheme without our heuristic, the performance would still be poor. It is the edges that are congestion-aware that provide the real additional benefit when traffic is non-homogeneous.

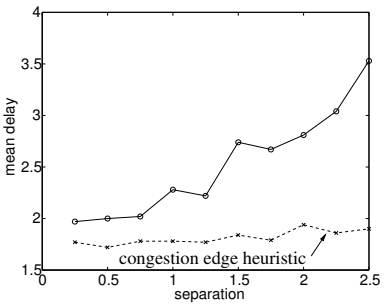

Fig. 9. Average traffic delay vs. dipole separation for $\gamma_{\text{non-uniform}} = 0.1$.

IV. Conclusion and ongoing work.

In this paper we focused on addressing the impact of non-homogeneity on the performance of a p2p architecture for storing spatio-temporal events. Non-homogeneity manifests itself in p2p networks not only in terms of the traffic but in terms of peer locations and thus topology. Peers with disproportionally large cells can become hotspots for the performance of the entire network. We presented mechanisms to augment the overlay topology with congestion driven edges and/or peer incentives to dynamically reconfigure the topology to achieve an even traffic distribution. In our ongoing work we are extending our platform's capabilities with a method to add fault-tolerance by replicating data to the peers closest to each event's associated location. Additionally, we are extending data management to address issues like load-balancing enabling, if needed, better sharing of storage resources when they are limited at each peer.

References

[1] M. Weiser, "The computer for the twenty-first century," *Scientific American*, pp. 94–10, September 1991.

[2] J. Ott *et al.*, "Floating content: Information sharing in urban areas," in *PerCom*, 2011.

[3] A. Ziotopoulos and G. de Veciana, "P2P network for storage and query of a spatio-temporal flow of events," in *PerCom Workshops*, 2011.

[4] B. Karthik *et al.*, "Load balancing in dynamic structured p2p systems," in *Proc. IEEE INFOCOM, Hong Kong*, 2004.

[5] D. Karger and M. Ruhl, "Simple efficient load balancing algorithms for peer-to-peer systems," in *Proc. of the 16th annual ACM symp. on Parallelism in algorithms and architectures*, 2004.

[6] K. Aberer *et al.*, "Multifaceted simultaneous load balancing in dht-based p2p systems: A new game with old balls and bins," in *Self-* Properties in Complex Information Systems, Hot Topics series, LNCS*. Springer, 2004.

[7] D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and its Applications*. Chichester: John Wiley and Sons, 1987.

[8] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. of the 32nd ACM Symp. Theory Computing*, 2000, pp. 163–170.