P2P Network for Storage and Query of a Spatio-Temporal Flow of Events

Ayis Ziotopoulos, Gustavo de Veciana

E-mail: ziotopou,gustavo@ece.utexas.edu

Abstract

We develop a novel approach to store and query real-time (contextual) information from/to a population of wireless/wired devices generating a stream of spatio-temporal events corresponding to sensors/applications and/or humans. The approach is based on a novel peer-to-peer overlay network architecture which exploits the spatial relevance of events and the locality of queries to store and query events close to where they are likely to be consumed. This paper focuses on the development of the basic protocols necessary to operate such a system. We propose a simple model for such systems and explore the sensitivity of performance (query delay) to node density and overlay connectivity.

I. Introduction

A significant and largely unaddressed challenge in the wide-spread use of networked systems is the harnessing of real-time flows of spatio-temporal information to deliver real-time context-awareness to applications and their users. In this paper, we study a novel peer-to-peer (p2p) infrastructure enabling wireless and wired users/devices and their associated applications to store and query a flow of events that are short-lived and associated with particular spatial locations. We focus on systems exhibiting locality in queries, i.e., queries are likely to be related to events located in the vicinity of the device/user making the query, see e.g.[1]. To motivate our design, consider a system for managing available parking spots. Primitive events signifying (possible) vacant parking spots are contributed by a disparate collection of devices/users including sensors, payment stations, cameras surveilling the streets or the human drivers who vacate a spot. A fixed peer overlay is used to store the corresponding events and answer queries about them from interested drivers. Such a system would be a representative application to use our proposed system.

This paper focuses on exploring the design space of overlay infrastructures of fixed peers that act as gateways for mobiles, this approach has already proved its merits, see, e.g., [2]. The challenge lies in designing a p2p network with the primary purpose of supporting spatio-temporal context-awareness based on a flow with high spatial intensity of many short lived events. In this context a simple very low overhead design which achieves low query delays is desirable. With our design we expect to achieve reduced traffic (due to the spatial locality of events and queries) and ultimately reduced query delay (since queries are resolved locally) in addition to inheriting the advantages of p2p architectures (e.g., distribution, scalability, robustness, lack of central administration, use of idling resources, enabling of grass roots deployments).

Related work. Computing context from primitive events is a problem of growing interest in the pervasive computing community. Known approaches based on storing events in p2p overlays and computing on the results of queries do not fully satisfy the requirements imposed by the real-time nature scenario we target in this paper. Seminal DHT-based p2p storage systems, e.g., Chord [3] Pastry [4] or GHT [5] are ideally suited for applications performing *location-independent* naming and load balancing across storage capacity, see [6]. They do not preserve locality in storage, i.e., two files contributed by the same peer are not necessarily stored at the same (or close-by) peers nor do they explicitly incorporate a notion of the stored information's relevance in space and time. Attempts to overcome these limitations while still remaining in the realm of DHT-based overlays do not fully succeed. Kuhn et. al. in [7], introduce "*containers*" as a new abstraction for storing spatially related events. This way of storing events lacks flexibility when compared to our approach and does not naturally maintain locality, i.e., containers have to be specified ahead of time and two events that are within a query-defined distance to each other are not guaranteed to lie in the same container, this will become obvious subsequently in this paper when we introduce the 'range' query. PeopleNet, [8], introduces the 'bazaar', a topic specific region to resolve topic-specific queries. Neither of these approaches preserves locality in storage.

Context can be computed and queries resolved more efficiently in a platform providing access to relations between events/info rather than simply to the individual

events/info. *Similarity* queries, see [9], are a reasonably powerful family of queries for computing context derived from events exhibiting locality, e.g., retrieving events tied to a given region. Resolving these types of queries calls for a modified p2p architecture directly factoring *locality* into information storage and retrieval. Voronoi tessellations have proven to be a useful geometric structure to capture similarity queries and spatial locality in networks. SWAM [9] and VoroNet [10] are designed to enable efficient processing of similarity queries by using Voronoi diagrams. The key idea is to embed data in a d-dimensional attribute space, and, based on an attribute distance metric, determine the induced Voronoi tessellation and Delaunay graph. Edges in the Delaunay graph (in the attribute space) are mapped to edges in the overlay network among the peers that currently hold the data items. By contrast, our focus is on the physical proximity of the locations to which events/information are tied and to their regions of relevance. Links are driven by the spatial proximity of peers and network performance concerns alone, rather than the data's attributes. Challenges for SWAM and VoroNet lie in the overheads associated with managing a d-dimensional Voronoi diagram particularly when peers join and leave the p2p network frequently. By contrast, in our work peer churn results in moving data among peers to maintain network functionality. Our work and these approaches consider the introduction of additional overlay edges, such as those proposed in [11], to improve query performance. However, in our work we study their effect on query delay rather than the hop-count and we show how such edges and topology should be managed to optimize system performance.

A platform for distributing the location updates for players in MMOGs is presented in [12]. Updates about a player's location need only be distributed to a neighborhood around the player. This is a form of locality in a *virtual* space as opposed to our work which employs locality in the *physical* space. This observation has critical implications on performance as physical locality correlates positively with low delays in the underlay network. The work in [12] uses geometrical routing on the overlay network as we do although the underlying topology is quite different from ours. Our work is based on a *generic* event model that takes explicitly into account spatio-temporal locality. No such model is offered in [12], where the focus is on a special type of information, i.e., location updates. In our work, events have a life-time that is independent of the life-time of the peers forming the overlay. In [12] the information disseminated, i.e., the location updates coincide with the life-time of the players in the MMOG. Moreover, in our platform a peer can store more than one event. In [12] information is not stored, each player knows its own location and publishes updates about it. Finally, in our work we offer procedures for storing, deleting and querying events. No equivalent procedures are offered in [12].

Contributions. In this paper we make several key contributions. First, we propose, to our knowledge, the first Voronoi-based p2p overlay network design to support storage and querying of a spatio-temporal flow of events/information whose consumers exhibit locality. We describe the protocols supporting our proposed architecture for topology and data management under peer churn. Second, we propose a grid network model including a stochastic spatio-temporal flow of events and queries, which enables a preliminary study of the network's performance characteristics. Specifically, we are able to explore the sensitivity of performance to peer node density and to the overlay connectivity among the peers, e.g., moving from neighbor connections only, to an overlay network augmented with Kleinberg edges [11]. These results are further validated via simulation over networks with randomly located peers, i.e., (as opposed to a grid.) Our main finding is that given an event and query load there exists a range of peer densities which offer good query delay - one would want to operate the system there. We also show the beneficial impact of the Kleinberg edges on the mean query delay, we suggest a further optimization which involves only placing Kleinberg edges on the scale of the locality of queries.

II. Event and Query Models

Our focus is on capturing a spatio-temporal flow of (short-lived) events which can represent a wide range of data/information associated with users, applications, sensors or machines. These are formally defined as follows:

Definition 1. *An event e is a five-tuple*

$$(e.location, e.range, e.time, e.duration, e.type) \in E$$

where $E = \mathbb{R}^2 \times \mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}_+ \times \mathbb{T}$ and $\mathbb{T}$ denotes a set of possible event types.

As shown in Fig. 1, this can be visualized as a cylinder 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 (see [13] for a similar concept 'Area Of Interest (AOI)), as well as a time at which it is generated/starts and duration: *e.time* and *e.duration* respectively. We assume that the devices/applications that generate events have geo-location capability and synchronized clocks. This allows them to 'stamp' the events with meaningful spatio-temporal coordinates. 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*. The possible event

types, e.g., 'change-of-status' type, are assumed to be predefined, but for the remainder of the paper we will not focus on this aspect.

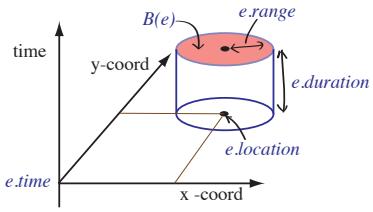

Fig. 1. Event model.

The goal of the infrastructure is to efficiently support spatio-temporal queries. In this paper we focus on *range queries* on *active* events. This is done to simplify notation, i.e., remove time, but one can easily generalize this to consider queries on past or even future events. Range queries are defined as follows:

Definition 2. *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. See [14] for definitions of more complex queries, e.g, the k-nearest neighbor query and how they can be expressed in terms of the range query.

We conclude by formalizing the operational scenario of interest in this paper, specifically:

Assumption 1. Spatial locality of events and queries. *Event and queries submitted by users/devices exhibit locality in that they are associated with close by locations.*

Although we will not preclude queries not satisfying such spatial locality, this assumption will drive in part our design choices. It is motivated by the idea that spatio-temporal context awareness for short lived events makes sense, and is most likely to be relevant, close to where users/devices currently dwell.

III. Architecture

We consider a platform where the participating entities can play one or more roles: contribute events, make queries, and/or serve as a peer in the p2p overlay network. Further we envisage a setting where there may be (mobile) wireless entities, e.g., sensors, phones etc., which can not serve as peers, and wired fixed devices, e.g., PCs and corporate servers, which can. All entities are assumed to know their locations, but exact locations are not necessary. Fig. 2 exhibits the elements of the platform which will be discussed below.

A. Overlay Topology Management & Routing

We let $\Pi = \{p_1, p_2, \ldots\}$ denote the set of peers identified by their unique locations $p_i \in \mathbb{R}^2$. They are

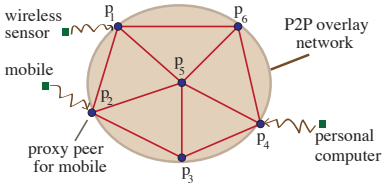

Fig. 2. Elements of the p2p architecture.

interconnected via links in a structured p2p overlay, represented in Fig. 2 by thick straight lines. Overlay links are realized by one or more physical links at the IP layer. The basic overlay connectivity is simple and driven by the goal of exploiting the spatial locality of queries – it corresponds to the Delaunay graph induced by the peers' locations. See [15] 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 $\Pi \subset \mathbb{R}^2$. In the sequel, when clear we will abuse notation and denote the Voronoi cell $C(p|\Pi)$ corresponding to the peer p by C_p. We let $N(p)$ denote the set of peer p's neighbors in the Delaunay graph.

There is already substantial work on distributed protocols to maintain a Delaunay graph among a set of peers, addressing efficiency, correctness, and, in part, peer churn, see e.g., [16; 17; 18] and [19]. This work is orthogonal to ours. As such, in this paper we will defer to this work, and simply assume that protocols are in place to maintain such overlay structures.

Assumption 2. ***Topology and Routing.*** *We assume the overlay connectivity of our platform is a* superset *of the Delaunay graph, and peers execute a distributed protocol to maintain the Voronoi tessellation and 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 the neighbor which is closest to l with the intent of eventually reaching the peer closest to the location. These choices for our platform result in the following beneficial properties:

Greedy routing always converges. Greedy routing on a Delaunay Graph always succeeds, see [20]. Moreover such routing on a superset of the DG also succeeds, see [21], thus we will consider additional overlay links to optimize system performance. Note that the length of the shortest path between two peers in the Delaunay graph is a constant times their Euclidean distance, see e.g., [22]. Thus, paths chosen on the overlay network might roughly correlate to low cost routes in the underlying IP network. This assertion is considered critical by Castro et. al. [23].

Limited routing storage overhead per peer. Greedy

routing requires each peer to maintain the location of its neighbors in the DG only, Moreover, the routing information exhibits locality, that is, inconsistencies in the routing information at a given peer will not lead to poor routing outside the peer's local region. We deem this important for networks with peer churn.

Maintaining the DG under churn is scalable. For p2p networks experiencing churn, some members of the network will have to update their neighborhoods. It can be proved that when a peer joins/leaves the p2p overlay, only its neighbors and on average an $O(1)$ additional set of peers must update their state – we omit the proof due to space restrictions, see also [24]. Thus topology management under churn in our platform is not computationally demanding and can be done in a distributed manner.

Routing on the overlay topology, assuming a homogeneous distribution of peers, roughly implies that the peer hop count will grow linearly in the distance to be traveled. Adding extra edges among peers, i.e., operating on a superset of the DG, can significantly reduce the number of hops a query has to traverse. This however requires adding edges with care. In the sequel we will use the well known result of [11] which we summarize as follows

Fact 1. *[11] Let s and t be the source and destination peers of a query respectively and suppose they are drawn uniformly from an $n \times n$ square grid. Suppose each peer u on the grid has edges to its immediate neighbors as well as to one other peer v chosen with probability proportional to $d(u,v)^{-2}$ where $d(u,v)$ denotes the distance between u and v, then there exists a decentralized algorithm performing greedy-routing and a constant c, independent of n, so that the expected number of hops between u and v is at most $c(\log(n))^2$.*

A mechanism to add such edges to our overlay network is easily devised. 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, and in §IV we explore the resulting performance benefits.

B. Data Management and Query Processing

Our spatial locality assumption on events and queries motivates the following rule.

Rule 1. Event storage and deletion. *An event e is stored at the overlay peer $p \in \Pi$ which is closest to $e.location$, i.e., such that $e.location \in C(p|\Pi)$.*

This rule is easily implemented in our framework by greedily routing and storing the event to the peer p closest to $e.location$. One can consider various policies for event deletion. Since in this paper we focus only on *active* events they are deleted once they expire. However one can envisage policies where events are only deleted when storage space runs out or borrowing from caching policies whereby least recently queried events are deleted when new ones are to be stored.

One of our design goals is to address the possibility of peer churn. Clearly this not only requires managing the overlay topology but also where events are stored. Each time a new peer p_{new} joins the p2p network a subset of the events currently stored in the p2p network may fall within its Voronoi cell, and according to Rule 1 those events should be moved to p_{new}. Data management under churn is scalable having low overheads, since only events in p_{new}'s neighborhood may need to be moved:

Fact 2. Scalability of data management. *If a new peer p_{new} joins the p2p network only events stored by neighboring peers $N(p_{new})$ in the updated topology may need to be moved. If a peer p_{leaves} leaves the network its events will be moved to its current neighbors $N(p_{leaves})$. When nodes' locations follow a spatial Poisson point process the average number of neighbors is 6.*

The proof of Fact 2 is intuitively clear and omitted due to space limitations. Since a peer will typically have a low number of neighbors, churn results in low overhead, irrespective of the total size of the network, assuring the scalability of our platform.

One can envisage *lazy* mechanisms to reduce the overheads associated with moving events, e.g., for very short lived events and queries responded to on a best effort basis, placing new events as appropriate at p_{new} may suffice.

The ability to process range queries, see §II, is key to our proposed solution. A range query RQ(l,r) initially issued by a peer or proxy peer (on behalf of another entity using the overlay) q, is greedily routed to the peer p closest to location l. When the query arrives at p the query resolution process is initiated. Specifically p checks if it has events in the range query's disc and forwards the query to its neighbors. Neighboring peers check if their cell overlaps with the range query's disc, if so they check for events and forward the query to their own neighbors. We focus on basic functionality here, but clearly a peer need not forward the query back to the peer which originally forwarded it, and it need not forward it on more than once to its neighbors. This recursive query propagation eventually stops if there are only a finite number of peers in the range query's disc. The peers involved with the processing of the query then collect the intermediate results and forward them to the (proxy) peer q which originated the query. The resolution process is depicted in Fig. 3. Note, that the reply to a query can be sent directly to the IP address of the originating (proxy) peer q, reducing traffic on the overlay network. Algorithm 1 addresses the

range query resolution.

Algorithm 1 Range Query Resolution

1: **Resolve Query**$(q, p, RQ(l, r))$**.** {resolves $RQ(l, r)$ on behalf of peer q at peer p. }
2: **if** $B(l, r) \cap C_p \neq \emptyset$ **then**
3: $response\_set = \emptyset$
4: **for** all events e at p s.t. $e.location \in B(l, r)$ **do**
5: $response\_set = response\_set \cup \{e\}$
6: **end for**
7: **for** $t \in N(p)$ **do**
8: $response\_set = response\_set \cup ResolveQuery(p, t, RQ(l, r))$
9: **end for**
10: send $response\_set$ to q
11: **end if**

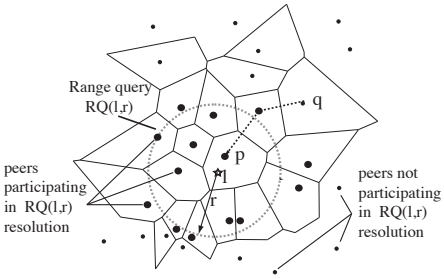

Fig. 3. Resolution of range query RQ(l, r).

IV. Performance Analysis

Query delays in our overlay network are to first-order determined by the number of peers a query message traverses and the queuing/processing at each intervening peer. A high density of peers can make the number of hops a query traverses high – this is a side-effect of our decision to route queries on the Delaunay graph. A low density of peers might result in peers being responsible for larger regions, increasing the storage and traffic loads they see, and thus increasing the per peer queuing delay. This suggests that for our network, "more is not always better." We explore this below.

Performance analysis. To study these performance trade-offs we consider the following *idealized* model. Assume that peers are spaced δ m apart in a square grid of *physical* dimensions $r \times r$ m^2, see Fig. 4, so there are a total of $\lfloor \frac{r}{\delta} \rfloor^2$ peers. To avoid edge effects we assume the grid wraps around, i.e, its geometry is akin to a torus, e.g., peer p_1 in Fig. 4 has a distance of 1 hop from the peers p_2, p_3 and p_4. Peers act as sources and destinations for queries. We assume that events are generated according to a spatio-temporal homogeneous Poisson Point Process (PPP) [25] with intensity γ_e events/sec-m^2. Each event is stored at its closest peer, so since each peer is associated with a cell of size δ^2 m^2, each peer sees an intensity of $\gamma_e \delta^2$ events/sec. Similarly, queries are submitted according to an independent spatio-temporal homogeneous PPP with intensity γ_q queries/sec-m^2. They are assumed to be processed by the closest (proxy) peer, so each such peer supports an intensity $\gamma_q \delta^2$ queries/sec. The destination of a query is *uniformly* distributed among the peers in a diamond with 'radius' l m centered around the associated source (proxy) peer. The parameter l is a measure of the spatial locality of the queries. The lower l, the higher the spatial locality. Queries are assumed to be greedily routed on the grid from the source peer to the destination, with ties broken at random. Queries are assumed to be point range queries, i.e., RQ(l, r) where r is negligible. To make things tractable we assume that the service time for processing, storing or relaying a query at a peer is an exponential with parameter μ. With these assumptions this model corresponds to a network of M/M/1 queues with generalized routing [26]. This grid model is only a first-order caricature of a homogeneous system in terms of both the traffic and peer topology, but as we will see gives some key insights on the characteristics of such systems.

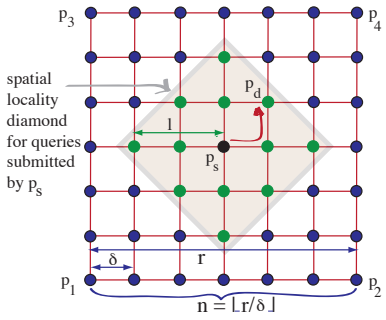

Fig. 4. Idealized grid model.

Symmetry in this grid model allows us to easily estimate the end-to-end delays on the overlay network. To account for the relayed traffic let $h(\delta)$ denote the average number of hops traversed by a typical query when the peers are spaced δ m apart. Then, the total traffic load that will be serviced in the network consists of two components: (1) storing of events, which is $n^2 \gamma_e \delta^2$; and (2) query relaying and processing, which is the number of peers n^2, times the query load per peer, times the average hop count, i.e., $n^2 \times \gamma_q \times \delta^2 \times \max(1, h(\delta))$. By symmetry, the total load is divided equally amongst the n^2 peers giving a load per peer of

$$\gamma(\delta) = \gamma_q \times \delta^2 \times \max(1, h(\delta)) + \gamma_e \delta^2. \quad (1)$$

The average end-to-end delay, $\overline{D}$, for a typical query corresponds to traversing $h(\delta)$ M/M/1 queues each supporting a traffic intensity $\gamma(\delta)$. Given the additivity of delays

experienced across the network we have that:

$$\overline{D} = \max(1, h(\delta)) \frac{1}{\mu - \gamma(\delta)}. \quad (2)$$

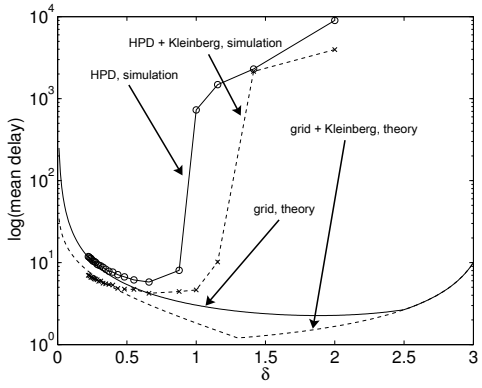

Fig. 5. Average query delay vs. cell side δ.

Note that topology and locality of the traffic on the overlay network impact the mean hop count $h(\delta)$ of a typical query. If each peer maintains edges to only its grid neighbors the number of hops a query traverses is roughly proportional to the mean distance and inversely proportional to δ, i.e.,

$$h(\delta) \approx c_1 \frac{l}{\delta}. \quad (3)$$

where c_1 is a constant. Fig. 5 exhibits a plot of the query's mean delay, i.e., Eq. 2, on a log scale versus the grid spacing δ when there is no locality, i.e., $l = r/2$, and $\mu = 1$, $r = 10$, $\gamma_q = 0.05$, $\gamma_e = 0.05$ and $c_1 = 0.5$. As can be seen a higher density of peers, i.e., lower δ, leads to a high mean query delay, due to the increased hop count of paths, while a lower density, i.e., high δ, leads to high mean query delays due to increased congestion in the traversed peers. Indeed, if δ is too high the traffic load on a peer may exceed its capacity. For a given set of system parameters there is an optimal density for peers, see Fig. 5.

Admittedly, the grid model is idealized. To capture more realistic topologies we simulated a Homogeneous Poisson Delaunay (HPD) topology. In a HPD topology the peers' locations are generated by a homogeneous Poisson spatial point process with intensity λ and peer connectivity follows the edges of the corresponding DG. The peers were placed inside a $r \times r$ square region and we used the same event/query model as the one described in the previous paragraph for the grid topology. Additionally, we consider a HPD topology augmented by Kleinberg edges. In Fig. 5, we include our simulated results for HPD topologies. We use the same horizontal axis for grid/HPD through the relationship $\delta \sim \frac{\sqrt{r^2}}{\sqrt{n^2}} \leftrightarrow \frac{1}{\sqrt{\lambda}}$. The theoretical results based on the grid capture the qualitative behavior of the HPD topology but there is a significant quantitative discrepancy for high δ/low λ. The latter is due to the arbitrary choice of the constants involved in Eqs. 3,4 and the effect of the statistical variations of the cells' sizes. Indeed, in contrast to the grid model, in a HPD topology the cell sizes for all the peers are not the same. This means that a peer with a bigger cell than the rest will suffer more traffic - that can lead to instability affecting all the queries routed through that peer. Thus, the effect we show for high δ/low λ. When λ is high the average cell size goes down and the effect of varying cell sizes is mitigated, this is a fundamental property of HPDs. In this case the traffic becomes more uniform essentially like a grid - observe the convergence of the theoretical and simulated curves. This motivates our future research on ways to adapt the topology to non-uniformities either by adapting the peers' cell sizes, e.g., by associating a "virtual" location with a peer, allowing modification of the associated Voronoi cells or the edges connecting the peers.

As mentioned in §III-A one can improve the performance in this system by including additional edges in the graph. In particular adding a single additional edge per peer as proposed by Kleinberg (see Fact 1) reduces the mean hop count to

$$h(\delta) \approx c_2 (\log(\frac{l}{\sqrt{2}\delta}))^2 \quad (4)$$

where c_2 is a constant. Note in Eq. 4 we have accounted for the fact that our queries are uniform in a locality diamond of radius l. The diamond centered at each peer includes roughly $2\frac{l^2}{\delta^2}$ peers, giving an 'equivalent' square grid to that considered by Kleinberg with $n = \sqrt{2}\frac{l}{\delta}$ and divided by a factor of 2 to account for distances in wrap around geometry. As expected and shown in Fig. 5 these new edges substantially reduce the mean end-to-end delay on the overlay – here we set $c_2 = 1$. Indeed the best performance is achieved with a much larger δ, i.e., one can get away with a much lower density of peers.

Fig. 6 shows some simulated results for the mean query delay on various overlay topologies as query locality is varied. We compare three topologies: Voronoi, Voronoi augmented with Kleinberg edges, and Voronoi augmented with Kleinberg edges restricted to the locality diamond of each peer. The values of the specific parameters used is $\mu = 1$, $\gamma_q = 0.1$, $\gamma_e = 0.05$, $r = 10$, $n^2 = 5000$. The simulation lasted for 100000 time units. From Fig. 6 it is clear that when the locality of the queries is high, e.g., 0.5, the delay of the queries is mostly due to the processing rather than queuing and the grid overlay performs well. However, as locality is reduced, i.e., l increases, the benefit of Kleinberg edges appears, with an improvement

in performance of up to 50% for this scenario. Further if the scale of locality were known up front, then limiting the edges within the locality diamond leads to a small yet consistent performance improvement, although simply adding edges throughout the whole network is robust and performs quite well. The graph also shows the performance curves associated with Eqs. 2, 3 and 4, where c_1 and c_2 were estimated by matching the value of the equations with the values reported by our simulations for $l = \frac{r}{2} = 5$. As can be seen the analytical and simulation results match very well. This validates the ability of our model to capture qualitatively the performance of the actual system.

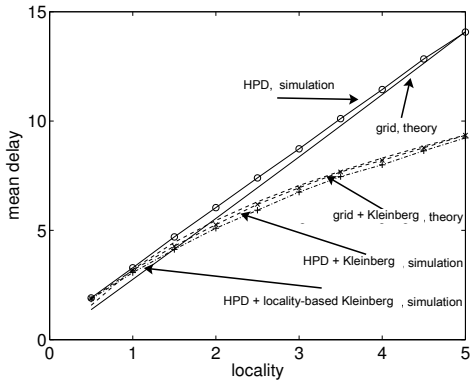

Fig. 6. Average query delay vs. locality.

V. Conclusion and Ongoing Work

We have introduced a novel p2p architecture for storing spatio-temporal events that is particularly suitable when there is spatial locality in queries. The core idea is to exploit spatial locality by aligning the overlay topology with the Delaunay Graph induced by peer locations, and augmenting it by Kleinberg edges. In our ongoing work we are extending our overlay'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. Ongoing work is also further addressing the impact of non-homogeneity on the performance of our network. 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 hot-spots for the performance of the entire network. Researching algorithms to augment the overlay topology with congestion driven edges and/or peer incentives to dynamically reconfigure the topology to achieve an even traffic distribution are also underway.

Acknowledgments

This work was partially supported by NSF Award CNS-0915928 and an AFOSR Award FA9550-07-1-0428.

References

[1] A. A. V. Castro *et al.*, "Hovering information - self-organising information that finds its own storage," in *SUTC*, 2008, pp. 193–200.
[2] N. Kotilainen *et al.*, "Mobile chedar - a peer-to-peer middleware for mobile devices," in *PerCom Workshops*, 2005, pp. 86–90.
[3] I. Stoica *et al.*, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM '01: Proc. of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2001, pp. 149–160.
[4] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *in MIDDLEWARE*, pp. 329–350, 2001.
[5] S. Ratnasamy *et al.*, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mobile Networks and Applications*, vol. 8, 2003.
[6] H. Balakrishnan *et al.*, "Looking up data in P2P systems," *Commun. ACM*, vol. 46, no. 2, pp. 43–48, 2003.
[7] E. Kühn *et al.*, "Integration of shareable containers with distributed hash tables for storage of structured and dynamic data," in *CISIS*, 2009, pp. 866–871.
[8] M. Motani *et al.*, "PeopleNet: engineering a wireless virtual social network," in *Proceedings of the 11th annual international conference on Mobile computing and networking*, ser. MobiCom '05, 2005, pp. 243–257.
[9] F. Banaei-Kashani and C. Shahabi, "SWAM: a family of access methods for similarity-search in peer-to-peer data networks," in *CIKM '04: Proc. of the 13th ACM international conference on information and knowledge management*. New York, NY, USA: ACM, 2004, pp. 304–313.
[10] O. Beaumont *et al.*, "Voronet: A scalable object network based on Voronoi tessellations," in *Proc. of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007*. Society Press, 2007.
[11] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. of the 32nd ACM Symp. Theory Computing*, 2000, pp. 163–170.
[12] B. Hariri *et al.*, "Using geometrical routing for overlay networking in MMOGs," *Multimedia Tools and Applications*, vol. 45, no. 1, pp. 61–81, 2009.
[13] S.-Y. Hu *et al.*, "VON: a scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, pp. 22–31, July 2006.
[14] A. Ziotopoulos and G. de Veciana, "P2P network for storage and query of a spatio-temporal flow of events," ECE Department, University of Texas at Austin, Tech. Rep., 2010. [Online]. Available: http:www.ece.utexas.edu/~gustavo/realTimeP2P.pdf
[15] D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and its Applications*. Chichester: John Wiley and Sons, 1987.
[16] D.-Y. Lee and S. S. Lam, "Efficient and accurate protocols for distributed delaunay triangulation under churn," in *ICNP '08: Proc. of the International Conference on Network Protocols*, 2008.
[17] J. Liebeherr *et al.*, "Application-layer multicasting with delaunay triangulation overlays," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1472–1488, 2001.
[18] G. Simon *et al.*, "Distributed dynamic Delaunay triangulation in d-dimensional spaces," Institut Eurecom, Tech. Rep., 2005.
[19] M. Ghaffari, B. Hariri, and S. Shirmohammadi, "A delaunay triangulation architecture supporting churn and user mobility in MMVEs," in *Proc. of the 18th inter. workshop on Network and operating systems support for digital audio and video*, ser. NOSSDAV '09, 2009.
[20] P. Bose and P. Morin, "Online routing in triangulations," in *ISAAC '99: Proc. of the 10th International Symposium on Algorithms and Computation*. London, UK: Springer-Verlag, 1999, pp. 113–122.
[21] M. Ghaffari *et al.*, "On the necessity of using Delaunay triangulation substrate in greedy routing based networks," *IEEE Communication Letters*, vol. 14, pp. 266–268, 2010.
[22] D. Dobkin *et al.*, "Delaunay graphs are almost as good as complete graphs," *Discrete Comput. Geom.*, vol. 5, no. 4, pp. 399–407, 1990.
[23] M. Castro *et al.*, "Future directions in distributed computing," A. Schiper *et al.*, Eds., 2003, ch. Topology-aware routing in structured peer-to-peer overlay networks, pp. 103–107.
[24] L. Guibas *et al.*, "Randomized incremental construction of Delaunay and Voronoi diagrams," in *Proc. of the 17th international colloquium on Automata, languages and programming*. New York, NY, USA: Springer-Verlag New York, Inc., 1990, pp. 414–431.
[25] J. Moller, *Lectures on Random Voronoi Tessellations*, ser. Lecture Notes in Statistics. Springer-Verlag, 1992, vol. 87.
[26] F. Kelly, *Reversibility and Stochastic Networks*. John Wiley and Sons, 1979.