International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 2, February - 2014

Data Consistency in Dynamic Data Dissemination Networks

Mahesh Gadiraju
Department of C.S.E.
S. R. K. R. Engineering College,
Bhimavaram, India

Valli Kumari Vastavayi
Department of C.S & S.T.
Andhra University College of Engineering,
Waltair, India

***Abstract*— Clients need consistent data in data intensive applications whether they are standalone or online applications. Content delivery networks are used to deliver the web content effectively from data sources to the clients. Previously, content delivery networks were used to provide static content. Now they are also used to deliver dynamic content. Dynamic data dissemination network is a hierarchical network of proxies used for disseminating dynamic data in content delivery networks. The main aim of dynamic data dissemination network is to provide consistent data to clients with high availability. High availability to clients is achieved by providing the data from proxies also known as data aggregators that are at the edge of the Internet closer to the clients. The data consistency is achieved by effectively propagating the updates of data from data sources to clients. In this paper we propose an optimal technique of maintaining data consistency called cooperative leases using application layer multicast with delaunay triangulation. The proposed technique reduces load on the server and is scalable than the other policies like Push, Pull, Push-or-Pull and Push-and-Pull policies.**

Keywords*— Content Delivery Networks; Data Aggregators; Delaunay Triangulation; Compass Routing; Application Layer Multicast.

I. INTRODUCTION

Caching is the most famous technique that improves performance of content distribution and delivery in WWW. Many caching techniques were evolved to overcome the limitation of the Internet while accessing Web content. As discussed in [1], most frequently used data is cached at the site infrastructure in backend caching. Since backend caches are at the site infrastructure itself, they are suitable for dynamic content also. As the content is far away from the clients, backend caches don't reduce network delays and load on the server.

Another approach of caching is proxy caching. In proxy based caching the content is cached outside the site infrastructure. The proxy servers are nearer to the clients at the edge of the Internet. On receiving the client's request, the proxy server checks to see whether the requested object is available with it or not. If the object is cached at the proxy, it sends the object to the client. Otherwise, it fetches the object from the source, sends it to the client and caches it for further use [2]. The content delivery networks (CDN) [3, 4, 5, 6 and 7] and dynamic data dissemination networks (DDDN) [8, 9 and 10] that use the proxy caching technique are discussed in the following subsections. Consistency maintenance of data in dynamic data dissemination networks is emphasized in this work.

A. *Content Delivery Networks*

Content distribution network or Content delivery network is an overlay network formed with collection of network elements distributed over the Internet. In CDNs content is replicated over many Web servers for effectively and transparently disseminating the content to the clients. The CDNs were evolved to overcome the limitations of the Internet when accessing Web content. They provide maximum bandwidth, high fidelity and high availability of data to clients by duplicating the content at the edge of the Internet. It also reduces server overload during flash crowd [11].

The CDNs which use edge server technologies like the Akamai and IBM edge server technologies provide better data services to clients and well suited for both static and dynamic content. But in these technologies maintaining coherence of copies of data distributed over different edge servers is difficult when the data is changing very frequently. Dynamic data dissemination networks are suitable for situations where data changes rapidly.

B. *Dynamic Data Dissemination Networks*

Dynamic data dissemination network is a CDN with hierarchical network of proxies used for effectively disseminating rapidly changing data. The proxies in dynamic data dissemination networks are called data aggregators. The data aggregators cache and disseminate the data effectively to clients. One or more data items are cached at a data aggregator required by the clients. The copies of same data item may be maintained at many data aggregators [12]. Since there are many copies of the same data item at different DAs, maintaining data consistency is an important issue in such networks. DDDN with sources (S1 and S2), clients (C1, C2 and C3) and hierarchical network of DAs is shown in Fig. 1. [11].

Whenever there is a change to a data item at the data source, the data item values at the DAs containing the data item also should be updated for maintaining consistency of data. For an ideal case the values of data items at the data aggregators and the value of data item at the source should be equal. For ascertaining this task every data change at the source should be propagated to each DA caching a copy of data item. It may not

be always possible to maintain such an ideal condition as it is not possible to propagate all the updates of rapidly changing data. Due to communication delays also it is not possible to maintain exact copies of a data item at the data source and at the DAs caching the data item. So, the copies of data are maintained satisfying the coherence requirements only. A client request includes coherence requirement of the requested data item[13].

The coherence requirement is the requirement for the allowable tolerance of deviation in the value of a data item available at client from its actual value at the source [14]. Coherence requirement may be time related value like the data item value is outdated for not more than a specified time interval. The requirement may be value related like the value of data item available at client should not differ from actual value of data item by more than some value called data incoherence. For example the client wants share price of a company in a stock market with a precision of one rupee/ dollar.

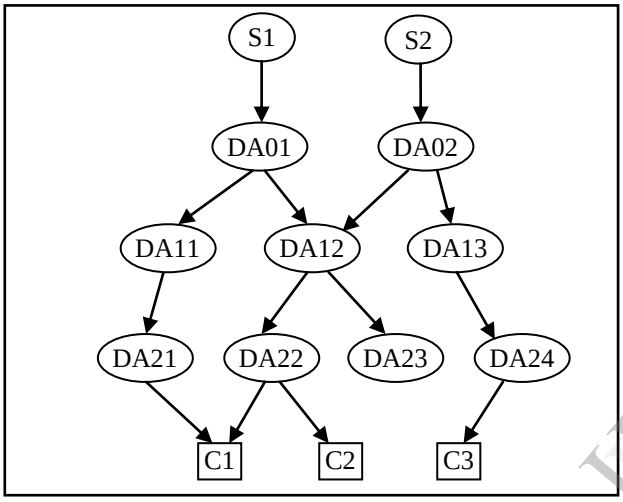

Fig. 1. Dynamic data dissemination network

II. DATA CONSISTENCY

There are different techniques of consistently updating the copies of data in dynamic data dissemination networks. The pull technique of maintaining data consistency is studied in [15] and the techniques like pull, push and combinations of push and pull are compared in [16 and 17]. The existing techniques of maintaining data consistency are given below.

1) Push technique
2) Pull technique
3) Push-or-pull technique
4) Push-and-pull technique
5) Cooperative Leases.

A. *Pull Technique*

In the pull technique of maintaining data consistency the clients poll the sources for knowing the data updates. When a data update of interest is there at the source, the client pulls such update. Time-to-live parameter attached with each object can be used by the client to poll for updates. When the Time-to-live for an object expires, the client polls the source for data updates. A high value of TTL minimizes the network overheads but may result in incoherent data. A low value of TTL increases the network overheads. So, TTL value should be computed carefully and it may be also necessary to update this value. Different approaches of estimating TTL value are given below.

1) Static TTL based on a priori assumptions
2) Semi-static TTL based on observed maximum rate of change
3) Dynamic TTL based on the most recent source changes
4) Dynamic TTL based on keeping TTL within static bounds
5) Adaptive approach

The static time-to-live technique involves in selecting a suitable constant value. This approach is simplest approach. But if the estimated TTL is high, the client may miss the required updates. If the estimated value is low, the client has to poll for updates more frequently. The semi-static approach is based on the maximum rate of change of data item. So, this approach has the disadvantage of unnecessary polls as the TTL is designated for worst case. Another approach is dynamic TTL based on the most recent source changes. The principle of dynamic TTL based on the most recent source changes is that the recent changes can be used to estimate the changes in the near future. Another approach is dynamic TTL based on keeping it within static bounds. The adaptive approach considers the effects of recent changes, static bounds and worst case of rate of change of TTL. The adaptive approach has the better performance than the other approaches. A parameter called time-to-refresh (TTR) based on coherence requirements is used in [17] for refreshing the data instead of TTL parameter. Pull technique has the advantage of high resilience to failures. But, it has lower degree of fidelity than the push technique.

B. *Push Technique*

In push based technique of maintaining coherency of data, the data updates are pushed from the data sources to data aggregators of the dynamic data dissemination network. Then the data aggregators disseminate the updated data to clients requiring the data. As discussed previously the clients specify the coherence requirement along with the data item requested. Here the main task is to disseminate the data items to the clients through data aggregators satisfying the coherence requirements. Each data aggregator may cache many data items and the same data item may be maintained at many data aggregators. The client selects one of the best data aggregator out of different data aggregators which provide the data item with required data coherence that is nearer to the client.

As discussed in [8 and 13], dynamic data dissemination network is used to disseminate the dynamic data and to propagate the data updates to DAs. Data dissemination graph is used in dynamic data dissemination network for propagating the data updates. Data dissemination graph is the union of dynamic data dissemination trees (DDDT) of all the data items. One data dissemination tree is constructed for each data item with the source of the data item as the root of tree and data aggregators caching the data item as nodes. Whenever a data

item is updated at the source, the update is pushed to the data aggregators of interest through the DDDT. The source that is the root node of the DDDT pushes the data update to its immediate child nodes or data aggregators. Each DA in turn pushes the data updates to its dependent DAs. A data update from a DA is sent to its child DA only if the update is necessary for maintaining data coherence at the child.

As discussed in [18], in each data dissemination tree of DDDNs, the values of data items have higher precision at parent data aggregators than that at child data aggregators That is the data incoherency bound at the parent DAs is less than that at child DAs. So, the data updates can be conveniently pushed from data source to the leaf nodes through DDDT from parent nodes to child nodes progressively.

Push technique can achieve 100% fidelity as the data is pushed from the source itself and it is aware of every data update. But this technique has lower resilience to failures than the pull technique. The pull technique has high resilience to failures as the updates are pulled from the clients. But the data updates at the source may be missed by clients as the pull technique is based on the estimated value of time-to-refresh or time-to-live. So, pull technique has less fidelity than push technique. The techniques that are combination of push and pull policies are discussed now.

C. Combinations of Push and Pull Techniques

The main principle of push-and-pull (PAP) technique [17] is that by default the DAs pull the data updates from the sources. But if some missing update is recognized by the server then it pushes the missed update to clients. Initially all the clients use pull connection and pulls the data items from the data sources according to the estimated value of time-to-refresh parameter. But if some data updates that violate the coherence requirements occur in between two pulls due to wrong TTR estimate, such updates are missed by the clients. To avoid such cases of missed updates in pull technique, the server also estimates TTR and knows the time at which the client is expected to pull the data. Server checks to see an update of interest to the client is not missed by the client. If a case of missed update occurs, the server starts sending the updates to the DAs and consequently to the clients.

For estimating the next pull of a client for an update, the server need not do complex calculation like clients do. It can compute TTR estimate based on the time taken for the last two pulls of the client. Let the source wants to compute next time (Ti) at which next pull occurs.

Let the time instances at which the two previous pulls occur be Ti-2 , Ti-1 and d be the duration of time required for previous consecutive polls i.e. Ti-1 - Ti-2. Equation (1) is used to calculate T_i by assuming a constant rate of change in data value.

$$Ti = Ti\text{-}1 + (Ti\text{-}1 - Ti\text{-}2) = Ti\text{-}1 + d \qquad (1)$$

If a data update at the server violating the coherence requirements occurs at a time instance $T < Ti$ then the server has to push such data update to the client. If all such updates are pushed then it is same as push policy. So, instead of pushing such updates the server may wait for a small amount of time e for the client to pull the data update. If e is zero then the PAP acts as push technique. But if e is more that is approaching the maximum TTR then PAP acts as pull technique. The performance of the PAP technique can be adjusted by varying the value of e. The advantage of PAP is that by pushing the missed updates from data sources it has more fidelity than pull technique. Another advantage is that it also improves TTR estimate there by improving the performance of pull technique.

The basic principle of push-or-pull technique [17] is to grant some clients push connections and the remaining clients pull connections. This categorization is done by the server depending on how many number of push connections it can serve. When a client requests a connection, if enough resources are available then the server grants push connection. If the resources are not sufficient for a push connection then the server has two options. First option is granting a pull connection. The second option is granting a push connection by dropping push connection already granted to other clients. The second option is used when the client requires high fidelity requirements. The candidates for dropping push connections are the clients with low fidelity requirement and those with high value of time-to-refresh values.

III. COOPERATIVE LEASES USING APPLICATION LAYER MULTICAST WITH DELAUNAY TRIANGULATION

Many techniques are there for propagation of data updates in content delivery networks. The different techniques in the literature [17] are pull, push, push-or-pull and combinations of push-and-pull. The push techniques have high fidelity and pull techniques have more resilience to failures. The cooperative leases approach proposed in [19] has the advantages of both push techniques and pull techniques along with scalability. This technique is based on the principle of leases for distributed file cache consistency proposed in [20]. In the leases technique of data updates a client or data aggregator makes an agreement with the server for notifying the data updates for a specified period of time. After the lease has been expired the DA may take the lease again. The lease includes the information like the leased data item (I) whose data updates should be pushed to the DA (D) and the lease period (T). A lease record is in the form {I, D, T}.

The leases approach has the server overhead of notifying all the updates to all the DAs having the lease. The cooperative leases reduces server load and are scalable than the other approaches. Instead of notifying all the updates in the lease period as in ordinary leases, the cooperative leases technique notifies updates only at every delta time units in the lease period. This type of relaxation in consistency maintenance is called delta consistency. In this technique client is never inconsistent by more than delta time units from its server version of the data. The delta consistency reduces server load of notifying all the updates

In the cooperative leases a single lease to a group of DAs is granted by server instead of granting a lease to every DA of the group that reduces the server overhead. The group lease is maintained by one DA of the group called leader. The server gives lease to leader and communicates with the leader only. The notifications to other DAs of the group are propagated by the leader. The cooperative leases record includes information like the leased data item (I) whose data updates should be pushed to the group (G), the leader of the group (L), the lease

period (T) and notification rate parameter Δ. A lease record in cooperative leases is in the form {I, G, L, T, Δ}.

In the event of first time request for a data item to a DA in a region, the DA decides the leader for the region. Then the DA sends HTTP request to the server and piggy backs leader information with the message. The message may include the notification rate parameter Δ. The server then sends requested data item to the DA and the lease information to the specified leader. Then onwards the leader is responsible for updating the leases and for notifying the data updates to different members of the group. The server has to notify the updates to the leader only. The group leases approach reduces the lease information to be maintained at server and reduces load on the server. The load of notifying data updates is transferred from server to the leader.

Whenever there are data updates at the server, the server notifies the updates to leaders of each group of interest at the notification rate (Δ) in the lease period. The leaders then propagate the notification of data update to the members of the group. In [19], application level multicast with hierarchy of DAs is used for propagating update notifications to DAs from the leader of the group that is also the root of the hierarchy. Here in this paper application layer multicast with delaunay triangulation is used to propagate update notifications from leader to data aggregators. Advantage of using delaunay triangulation for application layer multicast is that without any routing protocol, multicast tree can be embedded in the topology.

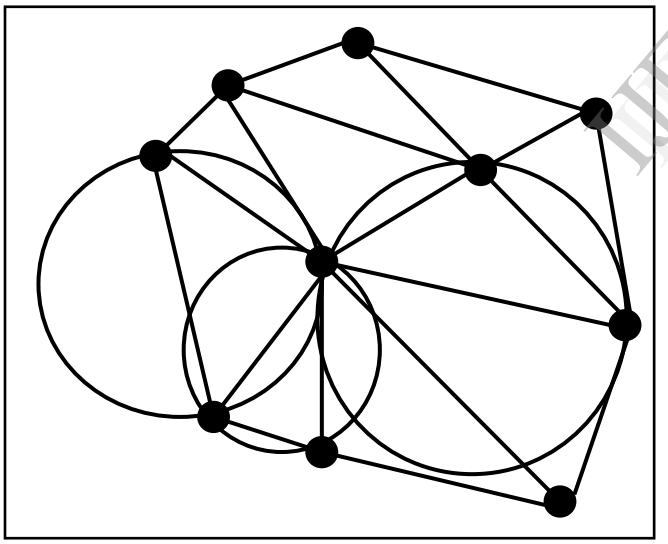

Fig. 2. Delaunay Triangulation of DAs in a group.

A triangulation of a set of given points in a plane satisfying the condition that no point in the set of points is inside the circumscribing circle of any triangle in the triangulation is known as delaunay triangulation[21]. An example for delaunay triangulation is illustrated in Fig. 2. The properties of delaunay triangulation are listed below.

- The exterior edges of the delaunay triangulation of a given set of points form convex hull for all the points in the set.
- Any two vertices in the delaunay triangulation are connected by an edge if only if there is an empty circle passing through the two vertices.
- The closest pairs of points in the given set points are neighbouring points in the delaunay triangulation.

Delaunay triangulation is constructed by considering a DA of each group as a node in the triangulation. The longitude and latitude of the data aggregators are assigned to the x, y coordinates of the corresponding nodes in the delaunay triangulation. The edges in the delaunay triangulation represent the logical links between the data aggregators. A delaunay triangulation for the group of DAs is constructed locally by keeping the triangulation equiangular. The meaning of equiangular triangulation is that the smallest angle of every triangle in equiangular triangulation is greater than or equal to optional triangles in other triangulations. It was shown in [22] that a locally equiangular triangulation of the convex hull of a finite set of points in a plane is delaunay triangulation.

A spanning tree with leader as root node embedded in the delaunay triangulation is used to propagate the update notifications. The route from the leader to a DA of the group is established using a compass routing [23]. The compass routing is a local routing algorithm and a node in the route between source and destination node requires position of source and positions of adjacent nodes only. Any node N can determine its parent node P for the sender S out of optional adjacent nodes such that angle between the lines NP and NS is least.

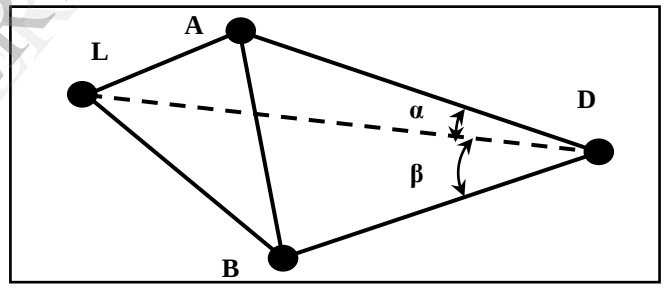

Fig. 3. Illustration of compass routing.

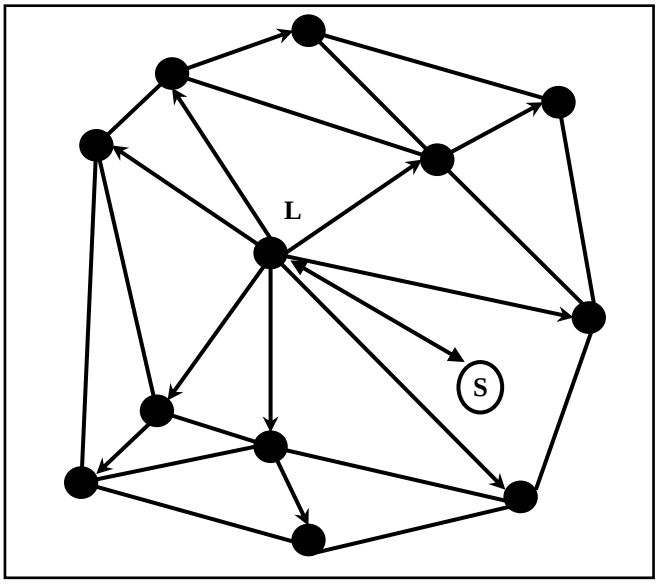

Fig. 4. Application layer multicast with delaunay triangulation.

The compass routing is illustrated in the Fig.3. For example the leader node is L and the node (or DA) whose parent node is to be established is D. Then by compass routing the node D selects A as parent because the angle between LA and LD is less than that between LB and LD ($\alpha < \beta$). The Fig. 4 shows the routes in which the data update notifications are multicast from the leader to different DAs. L is considered as the leader and S is considered as the data source in the Fig.4. On an update during the lease period, the update notifications are informed to the leader L and then the leader forwards the notifications to the members of the group in the routes shown by arrows. The advantage of using compass routing is that the route from the leader to each DA is computed locally at each intermediate DA. Another advantage is that the creation of triangulation doesn't require measurement between the DAs.

IV. Result Analysis

In this paper a data update propagation policy named Cooperative leases using application layer multicast with delaunay triangulation is proposed. In this policy application layer multicast with delaunay triangulation is used for propagating data update notification from group leader node to the member nodes of the group. Propagation of the updates through delaunay triangulation has the advantage of more resilience to failures than that through minimum spanning tree embedded in the group of nodes. These two approaches of multicast are compared for resilience to failures.

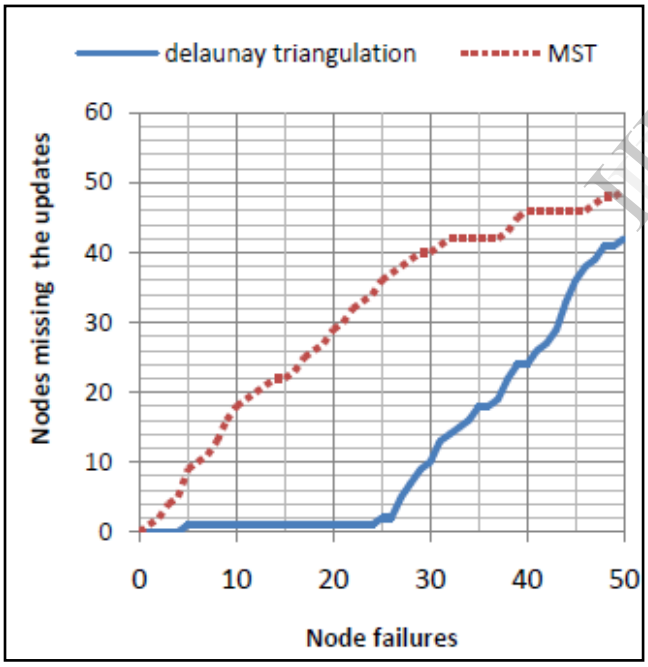

Fig. 5. Comparison of delaunay triangulation and MST

For comparing the two approaches, an experiment is conducted by taking a group with a size of 100 nodes. A random node is assumed to fail in each round of the experiment and observed the nodes/DAs that miss the data update notifications due to this failure assumption. The numbers of DAs that are assumed to fail are increased from 0 to 50 in steps of 1 and the DAs that miss the update notifications were observed for each failure. The results were plotted with number of DA/node failures on x-axis and number of DAs that miss the data updates on y-axis as shown in the Fig. 5. It was observed that the delaunay triangulation outperforms spanning tree for all the values of number of failure DAs

V. Conclusion

The problem under consideration is maintaining data consistency in dynamic data dissemination networks. The existing techniques for maintaining the data consistency are push, pull, push-and-pull, push-or-pull proposed in [17]. The main disadvantage of these techniques is scalability. This problem is solved by cooperative leases technique using application layer multicast proposed in [19]. But the disadvantage of the tree structure used in this technique for multicast is that the failure of a node in the tree causes a failure in a partition. To avoid such failures, cooperative leases using application layer multicast with delaunay triangulation is proposed in this paper. The results enhance the fact that the proposed approach is more resilient than the other approaches. It is also shown in this paper that how the leader of a group effectively multicasts data updates to DAs in the group using compass routing.

References

[1] A. Datta, K. Dutta, H. Thomas, D. Vander Meer and K. Ramamritham, "Proxy-based acceleration of dynamically generated content on the World Wide Web: An approach and implementation," ACM Transactions on Database Systems, Vol.29 (2) , pp. 403-443, 2004.

[2] M. Mahdavi, "Caching dynamic data for web applications, " Ph.D. Thesis, University of New South Wales, 2006.

[3] D. C. Verma, Content distribution networks: an engineering approach, John Wiley & Sons Inc., New York, USA, 2002.

[4] G. Peng, CDN: Content distribution network, technical report TR-125, Experimental Computer Systems Lab, Stony Brook University, 2003.

[5] M. Rabinovich Z. Xiao, F. Douglis and C. R. Kalmanek, "Moving edge-side includes to the real edge-the clients," In USENIX Symposium on Internet Technologies and Systems, 2003.

[6] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," IEEE Internet Computing, Vol.7 (6), pp. 68-74, 2003.

[7] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," Communications of the ACM , Vol.49(1), pp. 101-106, 2006.

[8] S. Shah, K. Ramamritham and P. Shenoy, "Maintaining coherency of dynamic data in cooperating repositories," In Proc. 28th International Conference on Very Large Data Bases, 2002, pp. 526-537.

[9] S. Shah, D. Shyamshankar and K. Ramamritham, "An efficient and resilient approach to filtering and disseminating streaming data," In Proc. 29th International Conference on Very Large Databases, Vol.29, 2003, pp. 57-68.

[10] R. Gupta and K. Ramamritham, "Optimized query planning of continuous aggregation queries in dynamic data dissemination networks," In Proc. 16th International Conference on World Wide Web, ACM, 2007, pp. 321-330.

[11] B. Rajkumar, M. Pathan, and A. Vakali, Content delivery network, Springer, 2008.

[12] G. Mahesh and V. Valli Kumari, "Primal-dual parallel algorithm for continuous aggregate query dissemination," In Proc. International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2013, pp. 545-549.

[13] G. Mahesh and V. Valli Kumari, "Optimal policy of data dissemination in CDNs," International Journal of Computer Applications, Vol.73, 2013.

[14] G. Mahesh and V. Valli Kumari, "Distribution of continuous queries over data aggregators in dynamic data dissemination networks," In Proc. Information and Communication Technologies, Springer Berlin Heidelberg, Vol.101, 2010, pp. 584-589.

[15] R. Srinivasan, Ch. Liang, and K. Ramamritham, "Maintaining temporal coherency of virtual data warehouses," In Proc. Real-Time Systems Symposium, 1998, pp. 60-70.

[16] P. Deolasee A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, "Dissemination of dynamic data," In Proc. ACM SIGMOD, Vol.30(2), 2001, pp. 599-599.

[17] P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham and P. Shenoy, "Adaptive push-pull: disseminating dynamic web data," In Proc. 10th international conference on World Wide Web, ACM, 2001, pp. 265-274

[18] S. Shah, K. Ramamritham and P. Shenoy, "Resilient and coherence preserving dissemination of dynamic data using cooperating peers," IEEE Transactions on Knowledge and Data Engineering, Vol.16 (7), pp.799-812, 2004.

[19] A. Ninan P. Kulkarni, P. Shenoy, K. Ramamritham, and R. Tewari., "Cooperative leases: Scalable consistency maintenance in content distribution networks," In Proc. 11th International Conference on World Wide Web, ACM, 2002, pp. 1-12.

[20] C. Gray and D. Cheriton, "Leases: An efficient fault-tolerant mechanism for distributed file cache consistency," In Proc. 12th Symposium on Operating Systems Principles, ACM, Vol.23 (5), 1989, pp. 202 – 210

[21] J. Liebeherr, M. Nahas and Si. Weisheng, "Application layer multicasting with delaunay triangulation overlays," IEEE Journal on Selected Areas in Communications, Vol.20 (8), pp. 1472-1488, 2002.

[22] R. Sibson, "Locally equiangular triangulations," The Computer Journal, Vol.21 (3) , pp. 243-245, 1978.

[23] E. Kranakis, H. Singh and U. Jorge, "Compass routing on geometric networks," In Proc. 11th Canadian Conference on Computational Geometry, Vancouver, Canada, 1999.