Data Aggregation Method for View Range Computation on P2P-based VCS

Ryo Nishide, Dai Ito, Masaaki Ohnishi, Shinichi Ueshima
Graduate School of Informatics, Kansai University
2-1-1 Ryozenji, Takatsuki, Osaka, 569-1095, Japan
Email: {fa4d003, fb6m124, fa4d001, ueshima}@edu.kansai-u.ac.jp

***Abstract*—Efficient data transfer is an essential topic to achieve scalability and data consistency to maintain the system in P2P-based Virtual Collaborative Space (VCS). In this VCS, each terminal requires the surrounding spatial data of its avatars for visualization of space. The congestion of avatars is then a serious problem when each terminal collects spatial data from the surrounding avatars. Thus, it requires a method to transfer data without delay and to relieve the load for terminals and networks. This paper proposes a data aggregation method on a P2P-based scalable geographic network to transfer the data efficiently at a congested area of avatars as nodes on a geographic network. The authors apply Skip Delaunay Network (SDN) generated from a hybrid structure of logical SkipNet and geographical Delaunay Network for remote access, and perform geocast for sending messages to a particular point or range on a plane. The authors conceive that multiple data paths to a common destination node construct a tree structure, in which the destination node is the root node, and nodes along the way are the internal nodes of the tree. Using the internal nodes for data aggregation, the proposed method can reduce frequent data transfer at a geographically crowded area of nodes. The authors show that data aggregation method on SDN can achieve both the long range contacts and reduction of CPU and network loads regardless of node distribution. The efficiency has been evaluated from the context of node congestion by examining the number of transferred data for methods with and without aggregation.**

I. Introduction

Virtual Collaborative Space System (VCS System), a system which uses the location and performs interaction on virtual space is gaining focus recently [1], [2]. It is a system with a set of interacting entities as avatars in virtual space. Users control these avatars from their terminals to walk-through in space, and perform interactions by sending messages to other users in virtual space.

Most of these VCS systems are built in C/S model [3], [4], which lacks scalability such as excessive cost for servers due to the increase of users. To overcome this problem, some efforts have been recently made for generating VCS systems on P2P setting [5–8], focusing on the characteristics as follows:

- Network scalability with respect to number of users
- System scalability according to spatial extension
- Distributive data management by space partitioning, and allotment of partitioned space to nodes

VCS systems have the same characteristic that each user requires only the local data of the surroundings. Thus, it is necessary to aggregate the surrounding data, and disseminate the data to a particular location or range on space. It is also necessary to cope with network congestion on P2P environment, by reducing inefficient data transfer for multihop communication.

We employ a well-known Delaunay diagram in computational geometry based on the adjacency of locations of avatars as nodes. We have proposed an autonomous and distributive generation algorithm of P2P Delaunay Network, which nodes generate such overlaid network cooperatively over 2D plane in P2P settings [9]. We have also proposed a Skip Delaunay Network (SDN) for reducing the number of hops for data transfer to remote nodes, using the hybrid structure of logical Skipnet and a geographical Delaunay network, and shown method to perform geocast to determine the directions to send data on geographical network [10].

If nodes within the view range increase, such problems as following become crucial in terms of network scalability, which requires a scheme to transfer data efficiently.

- Data Transfer Delay: Increase of number of hops
- Network Congestion: Data packet concentration to a specific peer

Furthermore, even though SDN can reduce the delay for data transfer, it cannot perform network load balancing for data packet concentration at a crowded area of nodes.

In this paper, we show data aggregation tree on SDN, which uses the scheme to temporarily cache multiple data and send them to the successor node at once. The paths for multiple data sent to the same destination node generate a tree structure, in which the root node is the destination, and each internal node of the tree is a node to cache multiple data and send them at once.

Using this scheme, we can reduce the frequency of data transfers between nodes, and cope with hotspots of nodes receiving data packets frequently. Our method employs advantageous features from both the SDN and aggregation method, by reducing the data transfer delay with SDN and handling network congestion with aggregation tree.

II. Features of Proposed Method

A. Generation of View

GUI Construction: In Virtual Collaborative Space, user's terminal requires the surrounding spatial data of its avatar to generate a view range for visualization of space. Thus, to construct a GUI, each user's terminal requires the spatial data managed by other users.

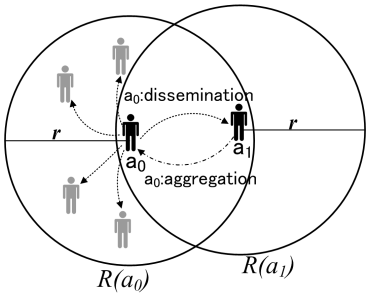
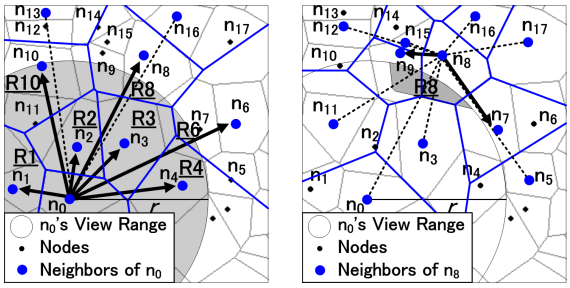


Fig. 1. a_0's Data Dissemination and Aggregation for Constructing a View



(a) n_0: Geocast in $R(n_0)$ (b) n_8: Geocast in R_8

Fig. 2. Geometric Routing to Nodes within View Range on SDN

To gather data from terminals of the surrounding avatars, it is necessary for other terminals to send their **(location, state)** data. On the other hand, it is also required to receive these data from the terminals of nearby avatars. Our model enables each terminal to generate a view with a certain range r by disseminating and aggregating these data to terminals of its surrounding avatars autonomously and distributively. Here, we assume the size and shape of view range is equal for every user's terminal.

A particular user's terminal which is controlling avatar a_i generates the view range $R(a_i)$, and sends data to every terminal which has avatar within $R(a_i)$. The number of terminals to send data depends on the density of avatars within the view range. Note that two particular avatars a_i, a_j mutually within their own view range require data of each other.

Fig. 1 shows an example for generating a view of avatar a_0. It shows that a_0 requires data of a_1 within the view range $R(a_0)$. On the other hand, a_1 requires a_0's data in the same way. Assuming that radius r of view range is the same with every avatar, a_0's terminal disseminates its data to its surrounding avatar's terminals, and aggregates other terminals' data within the view range $R(a_0)$ including a_1's data. In this way, the necessary spatial data are sent to all terminals with avatar in the view range.

Delaunay Network: Delaunay Network provides locality connections only with its adjacent neighbors, which can efficiently gather the surrounding spatial data required for generating a view. We use the locations of avatars controlled by users' terminals as nodes' location on Delaunay Network. Moreover, by assigning Voronoi Regions as the managing territories of space to the entire nodes, the entire data within the view range can be queried as the Voronoi regions of entire nodes cover the entire plane.

B. SDN as a Scalable Network

Geocasting within View Range on SDN

Delaunay Network requires a mechanism to access to remote nodes in case when node congestions occurred within the view range. Else, it can cause data transfer delay depending on the number of nodes within the view range. Thus, connections with remote nodes are necessary to send data within the congested area of nodes.

We use SDN, which is a hybrid structure of SkipNet and geographical Delaunay Network on a plane [10], for generating connections with remote nodes. Moreover, we perform geocast on SDN to send data to every node within the view range. Using SDN, we can build long range contacts (LRC) to send data to remote nodes, to deal with data transfer delay.

To perform geocast, each node generates Voronoi Diagram virtually with all of the neighbor nodes throughout the entire level of SDN. Using this Virtual Voronoi Diagram, we can determine the neighbor nodes to send data by extracting the intersection area A_i of query range and neighbor nodes' Virtual Voronoi Region. We send the data to every node, whose Virtual Voronoi Region intersects with the query range.

When data are sent to all neighbor nodes within view range, some nodes might receive the same data from multiple neighbor nodes. To avoid this, we set the direction to send data by replacing the query range with intersection area A_i, and send data with query range A_i to neighbor nodes. Thus, we can send data to a specific direction, and each query range data will be received only once.

We use the following notations to describe our method:

- NN: neighbor node set of the entire level of SDN
- $V.Vor(n_i)$: Virtual Voronoi Region of n_i
- $q(n_i)$: query range of n_i

Using these notations, we derive the following formula to determine the neighbor nodes $NN_{send}(n_i)$ to send data.

$$NN_{send}(n_i) = \{n_j \in NN | V.Vor(n_j) \cap q(n_i) \neq \phi\} \quad (1)$$

Instead of sending the view range to neighbor node n_j, we only send the intersection area $q(n_j) = \{V.Vor(n_j) \cap q(n_i)\}$ to node n_j. Note that this intersection area is used to set the direction to send data.

Here, we describe the method to perform geocast on SDN, to assure that n_0's data can be reached to every node within a certain view range r, using Fig. 2. Initially, n_0 generates $V.Vor$ with neighbor nodes of every level of SDN (blue dots on Fig. 2 left). In the figure, n_1, n_2, n_3, n_4, n_6, n_8, n_{10}, n_{13}, n_{16} are the neighbors of n_0. Among these neighbors, n_0 sends data to neighbors n_i with intersecting $V.Vor(n_i)$ and view range $R(n_0)$, which refers to all the neighbors of n_0 except n_{13} and n_{16}. The data includes the intersection area data R_i on the figure.

Then, node n_i which has received R_i data verifies their intersections with their neighbor's $V.Vor$ and R_i, and sends

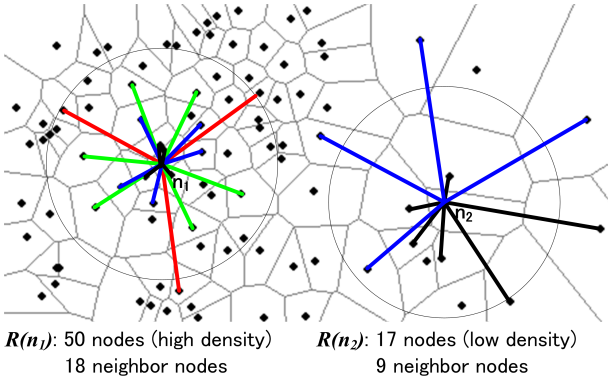


Fig. 3. Node Congestions and Number of Neighbors (Color = Level)

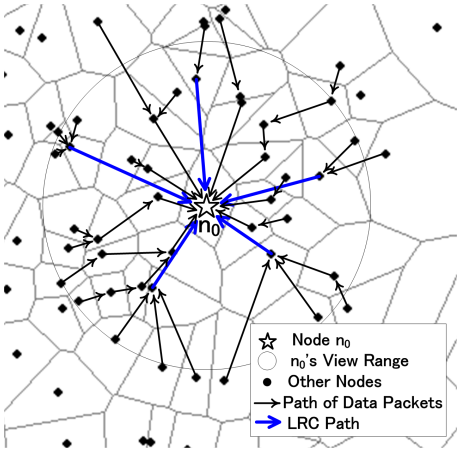


Fig. 4. Aggregation Tree Model for Data Transfer

intersection data to its neighbors in the same way. Fig. 2 right shows n_8, which has received R_8 data from n_0, sends the intersection data of neighbor's $V.Vor$ and R_8 to neighbor nodes n_7 and n_9, respectively. Using this routing scheme, the data can be delivered to the entire nodes within the circular range with radius r.

This method for disseminating data within view range has the following characteristics:

- Data can be sent to every node with an intersection of its Voronoi region and view range
- Data can reach the destination node(s) regardless of the location and shape of view range

Congested Nodes on SDN

In the previous section, we have shown that we can expect an efficient routing scheme for disseminating data to remote nodes by using SDN. We believe that this scheme can provide efficient routing for nodes within the view range. In this section, we describe how to solve the remaining problems for SDN when data concentrate to nodes in the crowded area.

In SDN, the network load can be balanced if nodes have equal size ranges and are uniformly distributed on a plane. However, when nodes are skewed at a particular location on space, the network load can be congested at the skewed location of nodes on space. Specifically, SDN has a characteristic that a particular node possesses many connections if a large number of nodes are within the view range, which increases the risk for data packets to be received from multiple nodes frequently (Fig. 3). Therefore, to deal with such problems, it is necessary to consider an efficient data transfer scheme to avoid network congestion.

When sending data packets, it is obvious that they should be sent with long messages up to the limit of window size, instead of sending numerous short message packets frequently. Otherwise, frequent transfers of short message packets can waste the bandwidth, as several packets may get lost due to buffer overflow, which requires the packets to be sent again. Resultantly, this increases the risk for massive packets to flow in the network, which causes network congestion.

Thus, in order to avoid such situations, we consider a model to cache multiple received data and send to the following neighbor nodes at once. In this way, we can save bandwidth by reducing frequent transfer of data packets, and reduce the number of data packets in the entire network.

C. Data Aggregation Method

In the previous section, we have shown method for nodes to send data using geocast, and explained that skewed distribution of nodes can cause network congestion when performing geocast on SDN. Here, we provide solution to reduce network load and frequent data transfer, using data aggregation tree to transfer data efficiently.

Data aggregation tree is a tree structure built from the paths of multiple data sent to a common destination node. This tree is generated passively from the paths of data sent by geocasting. The node which constructs the view range is the root node of the tree, and each node generates its unique data aggregation tree. Data are sent to its parent node of the tree recursively, until the data reach the destination node.

When data are sent to their parent node, multiple data can intersect at a particular internal node of the tree. From this internal node, the data take the common path to the destination node. We consider that instead of sending the data individually, multiple data should be cached and sent together as a single data to the successor node.

Here, we provide an example of data aggregation tree of node n_0 (Fig. 4). Let n_i, n_j be source nodes to send data d_i, d_j to destination node n_0 respectively. The paths for d_i, d_j have either of the following characteristics:

- d_i and d_j intersect at a particular node n_k, and takes the same path to n_0
- d_i and d_j intersect at n_0

In this tree, n_0 is the root node and intersecting node n_k is the internal node. To reduce frequent data transfer, d_i and d_j are packaged as a single data at n_k and sent to n_0 accordingly.

We believe that data aggregation tree can considerably reduce the data transfer frequency for these nodes. Moreover, the total data transfer frequency can stay low throughout the entire nodes.

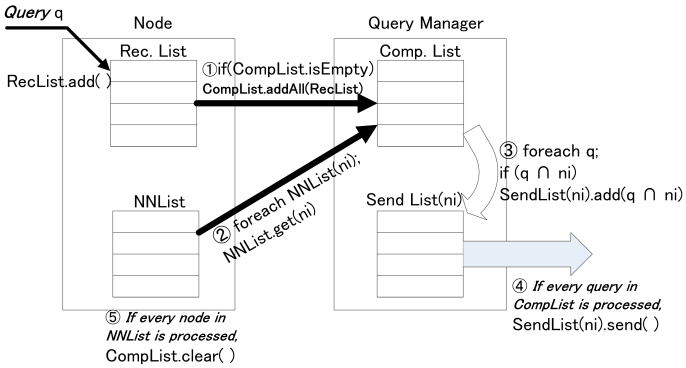


Fig. 5. Data Structure of our Method

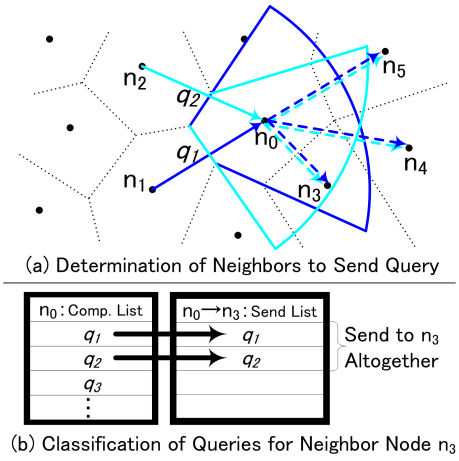


Fig. 6. Data Aggregation and Transfer Method

III. Proposed Method and its Data Structures

Here, we describe the proposed method for generating an aggregation tree. Figure 5 shows the data structure of our method and roles for each list. Initially, each node has three lists, namely Receive List (Rec List), Neighbor Node List (NN List), and Computation List (Comp List).

The queries include [Node ID, Packet ID, Location, Query Range, Time]. All the queries received from other nodes are stored in Rec List.

The details of the processes in Fig. 5 are as follows:

1) When Comp List is empty, move all the queries from Rec List to Comp List, and clear Rec List
2) Extract each neighbor node n_i from NN List
3) Obtain the intersection area with each query range in Comp List and $V.Vor(n_i)$. Generate Send List(n_i) and store intersection area for each query
4) When every query in Comp List is processed, send Send List(n_i) to node n_i
5) Perform step 2) with next neighbor in NN List. Clear the Comp List when every node in NN List is processed

Get Queries from Rec List

In order to determine the next destination node to send range query, we obtain the intersection area of query range and $V.Vor$ of neighbor nodes. We do not use queries in Rec List to obtain the intersection area, as this process should not be interfered by the process of adding queries in Rec List. That is, each node in NN List in turns obtains an intersection area with each query in Comp List, hence the new added query might not be processed by some neighbor nodes, which have already completed their intersection area computation.

Therefore, we use Rec List to store queries received from other nodes, and Comp List to get queries from Rec List. In detail, the Comp List pulls out a set of queries from Rec List and stores in Comp List, when the Comp List is empty. Moreover, the Comp List clears the list every time when the processes have been completed throughout all neighbor nodes.

Selecting Neighbor Nodes to Send Queries

Here, we describe the process to choose the neighbor node to send the queries in Comp List. To choose the neighbor node, the following information is required.

- $V.Vor$ of every neighbor node (in NN List)
- Query range from predecessor node (in Comp List)

To choose the destination node, we obtain the intersecting area of query range and $V.Vor$ for every neighbor node in NN List. Every neighbor node with an intersection area will be the target node to send the query. Instead of sending the view range as query range, we send the intersection area as query range to successor neighbor nodes.

Figure 6 (a) illustrates q_1 and q_2's query ranges intersecting with $\{V.Vor(n_3),\ V.Vor(n_4),\ V.Vor(n_5)\}$. Thus, the next destination nodes for q_1 and q_2 will be n_3, n_4, and n_5. Consequently, the next destination nodes (n_i) will be sent with the intersection area of $V.Vor(n_i)$ and q_1, q_2 respectively.

Send List for Packaging and Sending Queries

When sending packets to a particular neighbor node, it is inefficient to send range query packets one after another. Therefore, we generate a Send List to store every query with an intersection area of neighbor node, and send the queries together to the neighbor node.

The Send List is generated separately for each neighbor node, and each query in Comp List is classified to the Send List of relevant neighbor nodes. To send queries in Send List together, we generate a package with all queries in Send List, and send it to the neighbor node. In this way, we can send multiple queries together to neighbor nodes, avoiding frequent data transfer.

Figure 6 (b) shows an example for generating Send List of n_3, and utilizing it for packaging multiple queries to send to neighbor node n_3. Multiple queries q_1, q_2, q_3, ..., q_i are stored in Comp List. The Send List for n_3 stores q_1 and q_2 as these two queries are to be sent to n_3. Finally, q_1 and q_2 are packaged and sent to n_3.

IV. Evaluation

In this section, we verify the efficiency of our data aggregation method from the effects on node congestion within the view range. We have obtained the number of received queries according to the increase of nodes within the view range.

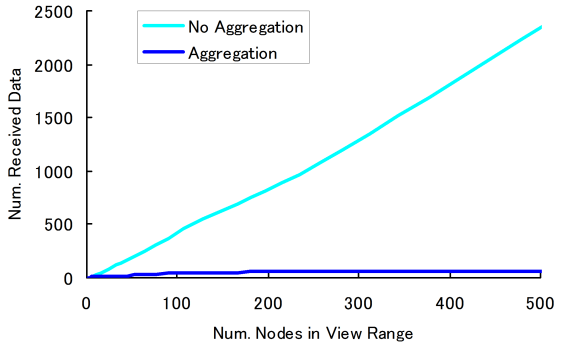


Fig. 7. Node Density and Number of Received Data

In our method, each node has a view range, and multicasts data within the view range. We fix the shape and size of view range for every node, and examine how the node density affects the CPU load for method with/without aggregation.

Settings for Simulation

Under the following settings, we have obtained the number of received data for with/without the data aggregation on SDN.

- **View range**: Nodes have circular view range
- **Node density**: Nodes increase within the view range
- **Data transfer**: Nodes send each data per step
- **Aggregation**: Package data and send to neighbor nodes
- **No aggregation**: Just cast data to neighbor nodes

CPU Load w.r.t. Node Congestion

We compare the method with/without aggregation in terms of congestion of nodes within the view range. In fig. 7, we examine the number of received data w.r.t. the number of nodes within the view range.

The figure shows that number of received data without the aggregation method rises steeply according to nodes increase. From the result, we consider that few increase of nodes can extremely increase the number of received data, as the data sent from a single node can be received from multiple neighbor nodes. On the other hand, the number of received data for aggregation method rises slightly, as multiple data are packed together and sent to the successor nodes. Therefore, the number of received data for aggregation method stays low even in node congestion.

We have performed step-by-step data transfer in this simulation. If we apply our method in real environment, we need to consider the CPU performance and data transfer speed. Thus, it is important to determine an appropriate interval time length for aggregating data, while considering the required threshold of time length for visualization and interaction in space.

V. Related Works

To obtain scalability in respect to number of users, a method to transfer spatial data efficiently is an important issue. Efficient routing mechanism and a scheme to reduce frequent data transfer are some required issues for achieving scalability.

For an efficient routing mechanism, some works on geocast have been proposed. [12] proposes a geocast routing mechanism based on node's location to send messages to nodes within a specified geographical area on mobile ad-hoc networks. [10] proposes method to construct a probabilistic link structure with remote nodes on P2P Delaunay Network, and applies geocast routing mechanism for sending messages to a specific geographical point or range on space.

For reduction of data transfer frequency, a method has also been proposed to construct a tree for aggregating and sending multiple data to its connected node. [13] proposes directed diffusion scheme for data dissemination, to intentionally aggregate multiple data at internal nodes of the tree. On the contrary, our method generates an aggregation tree passively from the path of multiple data sent using geocast.

In our proposed method, we can achieve efficient data transfer from both advantages of an efficient geocast routing mechanism and data aggregation method.

VI. Conclusion

We have proposed method to utilize Data Aggregation Tree for efficient data transfer, and examined that our method works efficiently on SDN through numerical simulation. Furthermore, with Data Aggregation Tree and geometric routing on SDN, we can perform geocast efficiently while avoiding network congestion.

For our plans in future works, additional evaluations are required with/without aggregation method, such as verifying the CPU and network load, acquiring the appropriate interval time lengths for aggregating data, and examining the amount of data loss due to transfer frequency of packets.

References

[1] B. Damer, "Meeting in the ether," ACM interactions, Vol.14 No.5, pp.16–18, 2007.
[2] M. Macedonia, "Generation 3D: Living in Virtual Worlds," IEEE Computer, Vol.40 No.10, pp. 99–101, 2007.
[3] Second Life, http://secondlife.com/
[4] Active Worlds, http://www.activeworlds.com/
[5] S.-Y. Hu, J.-F. Chen and T.-H. Chen, "VON: A scalable peer-to-peer network for virtual environments," IEEE Network, Vol.20 No.4, pp.22–31, 2006.
[6] B. Knutsson, H. Lu, W. Xu, B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games," In Joint Conf. IEEE Computer and Communications Societies, Vol.1, pp.107, 2004.
[7] Y. Kawahara, H. Morikawa, T. Aoyama, "A Peer-to-Peer Message Exchange Scheme for Large Scale Networked Virtual Environments," IEEE ICCS, pp. 957-961, 2002.
[8] P. Morillo, J.M. Orduña, M. Fernández, J. Duato, "Improving the Performance of Distributed Virtual Environment Systems," IEEE Trans. on Parallel and Distributed Systems, Vol.16 No.7, pp. 637-649, 2005.
[9] M. Ohnishi, R. Nishide, S. Ueshima, "Incremental construction of delaunay overlaid network for virtual collaborative space," 3-rd Proc. Conf. on Creating, Connecting and Collaborating through Computing (C5'05), (IEEE CS Press), pp.77–84, 2005.
[10] S. Tsuboi, T. Oku, M. Ohnishi, S. Ueshima, "Generating Skip Delaunay Network for P2P Geocasting," 3-rd Proc. Conf. on Creating, Connecting and Collaborating through Computing (C5'08), (IEEE CS Press), 2008.
[11] N. J. Harvey, M. B. Jones, S. Saroiu, M. Theimer, A. Wolman, "SkipNet: A Scalable Overlay Network with Practical Locality Properties", 4th USENIX Symp. on Internet Technologies and Systems (USITS '03), Vol.4, p.9, 2003.
[12] Y. B. Ko, N. H. Vaidya, "Flooding-based geocasting protocols for mobile ad hoc networks", Mobile Networks and Applications, Vol.7 No.6, pp.471–480, 2002.
[13] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks", 22nd Int'l IEEE Conf on Distributed Computing Systems, pp.457–458, 2002.