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

Graduate School of Informatics,
Kansai University

Ryo Nishide, Dai Ito,

Masaaki Ohnishi, Shinichi Ueshima

Table of Contents

Introduction

- **Virtual Collaborative Space (VCS) System is gaining focus recently**

  B. Damer, *ACM interaction*, 2007

  - **System with a set of interactive entities as avatars on virtual space**
    - **Users control their avatars from terminals for walk-through in space**
    - **Users perform interactions by sending messages to other users in virtual space**

Examples of VCS Systems:

Second Life (Linden Lab)

Digital Campus (Our project)

Recent Works on VCS Systems

- Most VCS systems built in C/S model
- C/S-based VCS systems **lack Scalability**
  - excessive cost for servers due to the increase of users

↓

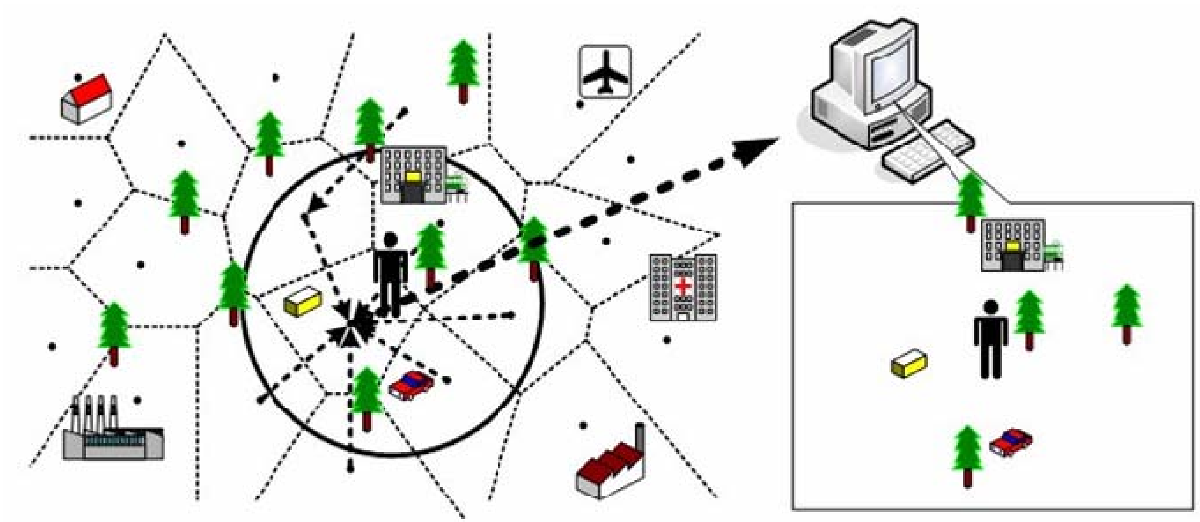
- Recently, some works on P2P-based VCS systems have been introduced

> **Related Works**
> **S.-Y. Hu, et al., *IEEE Network*, 2006**
> **B. Knutsson, et al., *IEEE Comp. and Comm. Societies*, 2004**
> **Y. Kawahara, et al., *IEEE ICCS*, 2002**
> **S. Rueda, et al., *IEEE VR Conference*, 2007**

  - These works focus on…
    - **Network scalability** with respect to number of users
    - **System scalability** for system maintenance or spatial extension
    - **Distributive data management** by space partitioning, and allotment of partitioned space to nodes
  - We focus on system scalability

Characteristics for VCS Systems

- Common Characteristics for constructing a GUI
  - Each user **requires only the local data** of its avatar's surroundings to generate a view
- Types of data streaming on VCS:
  - Video/audio live streaming
  - Avatar/object data streaming ← **We focus on these data**
    - **Avatar/object location, movement, action: *(location, state)*** data

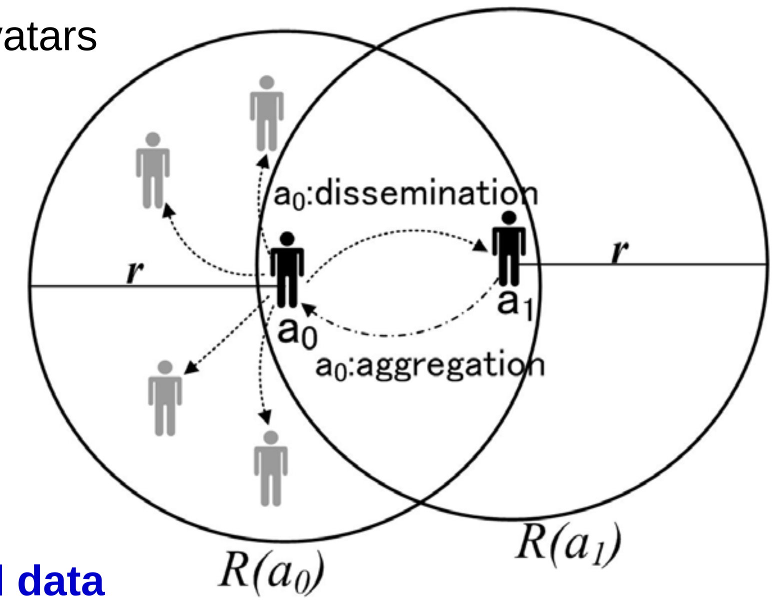


Aggregation of Local Data for P2P-based VCS

- In P2P system, **each terminal aggregates required data from the surroundings** to construct its own GUI
  - For C/S system, server provides necessary local data for each client individually
- Necessary to generate a view by **disseminating/aggregating** the spatial data to/from terminals of surrounding avatars

- Processes of user's terminal
  - Other terminals to send their **(location, state)** data
  - To receive data from the terminals of nearby avatars

- Notations for right figure:
  - $\boldsymbol{a_i}$: Avatar $\boldsymbol{a_i}$
  - $\boldsymbol{R(a_i)}$: View range of avatar $\boldsymbol{a_i}$
  - $\boldsymbol{r}$: radius of view range (Common size, shape)
- $\boldsymbol{a_0}$, $\boldsymbol{a_1}$ in the view range of each other
  - **a_0, a_1 requires data of each other mutually**

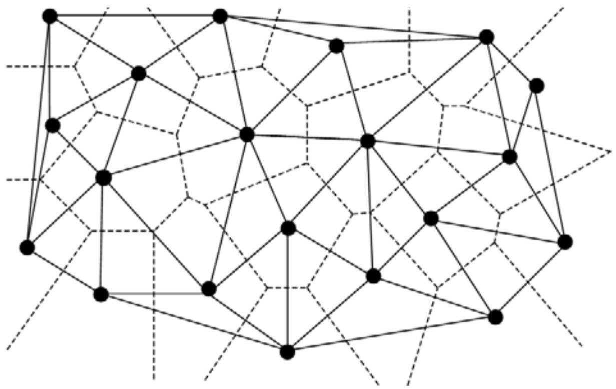


- **Resultantly,**
  - **Each terminal can obtain its necessary local data**
  - **Data can be delivered to nodes without requesting them**

Our Previous Works

- Our past proposals
  i. 3D VCS Prototype: **Digital Campus** (C5 2004)
  ii. Incremental construction of **P2P Delaunay Network** for VCS (C5 2005)
    - Employ a well-known Delaunay Diagram in computational geometry based on the **adjacency of avatars' locations**
      - Nodes: avatars controlled by users' terminals
    - Characteristics
      - **Autonomous and Distributive Generation:** Nodes cooperatively generate Delaunay Network autonomously and distributively
      - **Low Degree:** Average number of node degree is approx. 6
      - **Locality connections:** Nodes are connected locally.
        (Easy to aggregate data from surrounding nodes)

Digital Campus

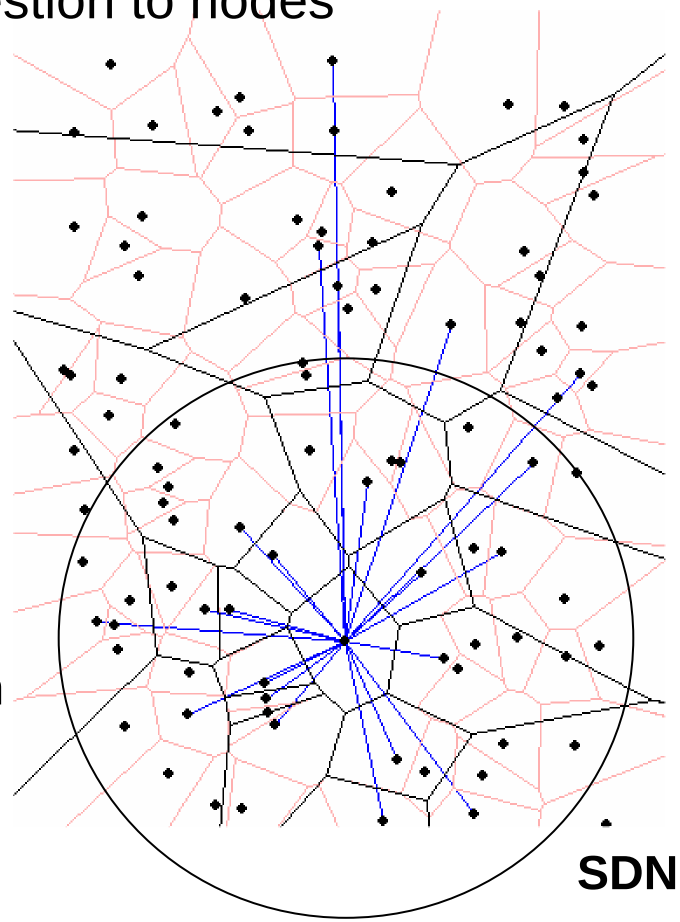


P2P Delaunay Network

Problems for P2P Delaunay Network

- Nodes within the view range increase
  - **Data transfer delay**: Increase of number of hops
  - **Network congestion**: Data congestion to nodes

- Combination of two methods as a solution
  - Data Transfer delay
    - Probabilistic link structure **Skip Delaunay Network (SDN)** for remote access (C5 2008)
  - Network Congestion
    - **Our proposed data aggregation tree** to reduce network congestion

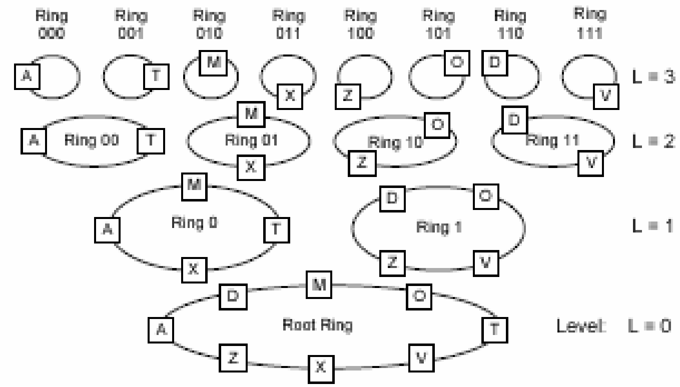


SDN

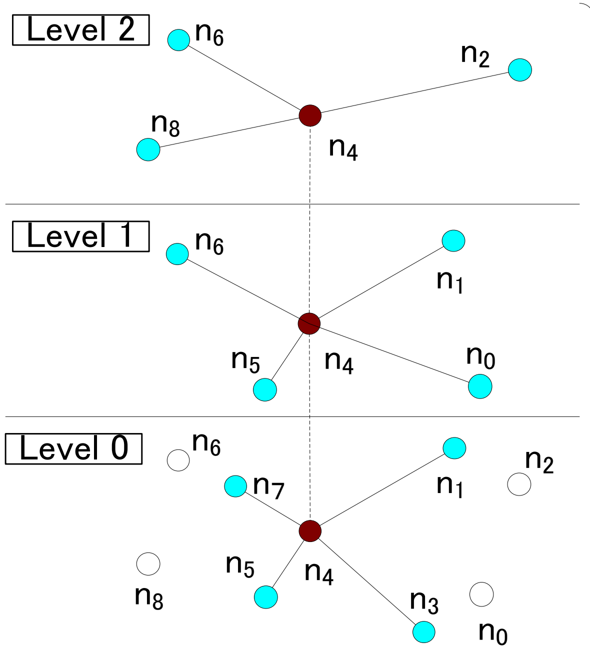
Skip Delaunay Network (SDN)

- SDN (Proposed in C5 2008)
  - Hybrid Structure: (SkipNet + Delaunay Network)
- Features
  - Autonomous and distributive generation
  - Routing scheme for efficient geocasting
  - Robust for rebuilding network when nodes join/leave
  - Low degree

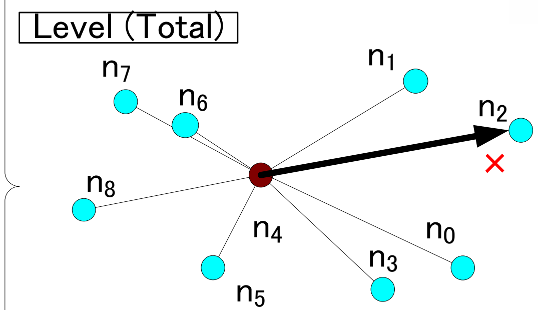
SkipNet (N.J.Harvey, 2003)



SDN

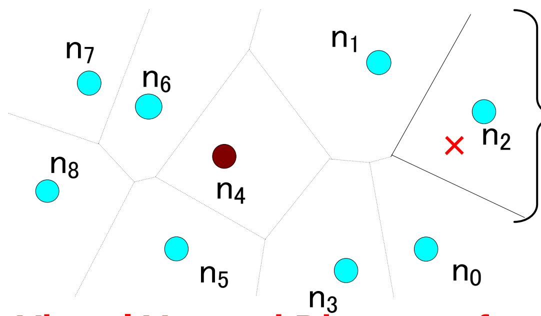


Base Delaunay Connections of n_4



× Destination

Level	n_4 Neighbor Node
0	n_1, n_3, n_5, n_7
1	n_0, n_1, n_5, n_6
2	n_2, n_6, n_8
Total	$n_0, n_1, n_2, n_3, n_5, n_6, n_7, n_8$



Virtual Voronoi Region n_2
$V.Vor(\boldsymbol{n_2})$

Destination Point within $V.Vor(\boldsymbol{n_2})$

Virtual Voronoi Diagram of n_4
(Generated from neighbors of entire level of SDN)

Geocasting on SDN

- **Geocasting on SDN**
  - Data distribution to nodes within the view range
  - **Geocast** for **unidirectional routing**

> **Notations**
> ***NN***: Neighbor node set of entire SDN
> ***V.Vor(n_i)***: Virtual Voronoi Region n_i
> ***q(n_i)***: Query range of n_i

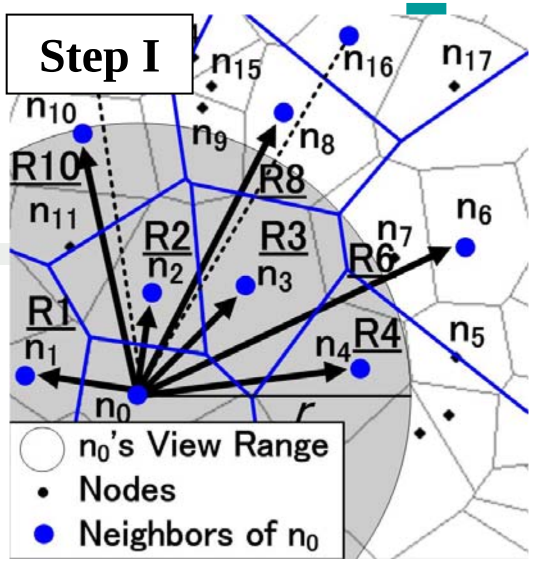
- **Processes**
  - n_i : Send data to node set $NN_{send}(n_i)$

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

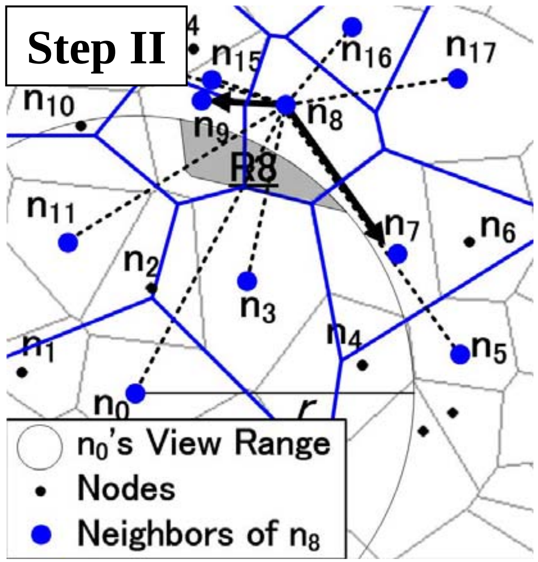
  - n_i : Data to send to neighbor n_j

    $$q(n_j) = \{V.Vor(n_j) \cap \bar{q}(n_i)\}$$

- **Advantages**
  - Data reachable **with few hops**
  - **Avoid redundant data transfer**
    - Data sent in a specific direction



n_0: Geocasting to R(n_0)

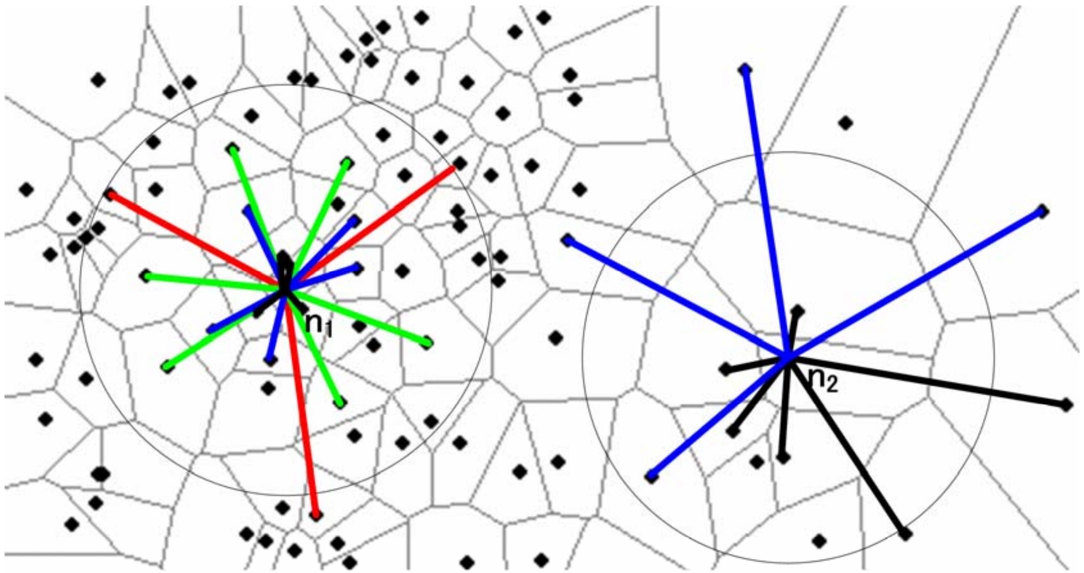


n_8: Geocasting to R8

Network Congestion

- Skewed node/network load at a **congested area of nodes**

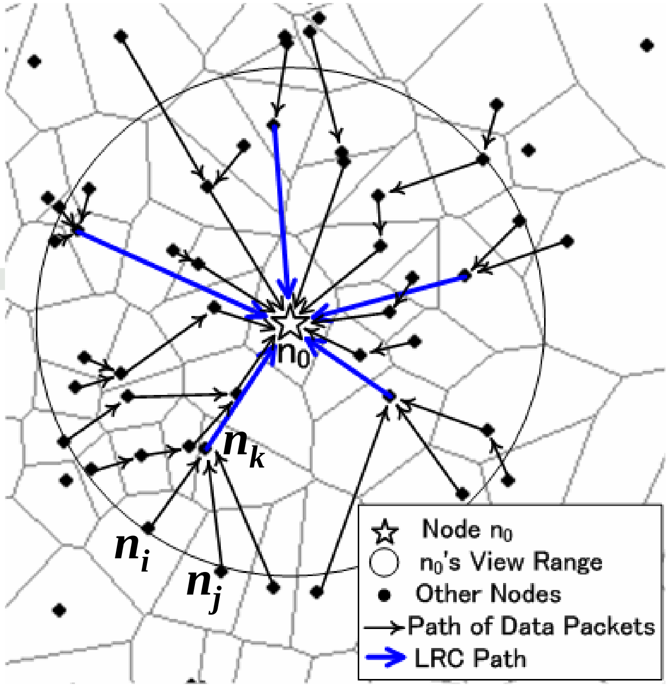
- Numerous nodes within the view range
  - Data disseminated to large number of nodes
    - Every node sends data to many neighbor nodes within the view range
    - Node receives large number of data from many neighbor nodes
  - If nodes send their streaming data continuously, it will be a critical problem in terms of node/network load



$R(n_1)$: 50 nodes (high density) 18 neighbor nodes

$R(n_2)$: 17 nodes (low density) 9 neighbor nodes

Our Idea of Data Aggregation



- **Data Aggregation Tree**
  - Tree structure built **passively** from data paths sent to same destination node
- **Notations**
  - n_i: **source node** to send **data** d_i to the same destination
  - n_0: **root node** to construct a view range (Each node with a unique tree)
- **Example:** Nodes **(n_i, n_j)** send data **(d_i, d_j)** respectively
  - d_i, d_j intersect at particular **internal node** n_k, and take the same path to destination
  - It is **inefficient to send multiple data (d_i, d_j, …) to the same destination separately**

  ⬇

  - We **package multiple data (d_i, d_j, …) as a single data at internal node n_k**, and send to destination node n_0 together

Features of Our Method

- CPU load reduction
  - Avoid nodes to receive data frequently
  - CPU load balancing regardless of node density and distribution types (uniformed, skewed)
- Avoidance of network congestion
  - Reduce network congestion and frequent packet flows w.r.t. node density
    - Packets are to be sent with long messages up to the limit of window size
      - Sending numerous short messages can cause packet loss due to buffer overflow
      - Resending lost packets can increase network congestion

Method for Constructing Aggregation Tree

- **Requirements for generating an aggregation tree**
  - **Caching scheme** of multiple data
  - Classification of multiple data to neighbor nodes
    - Package data to send to a common destination node
  - **Interval time** for storing/sending multiple data

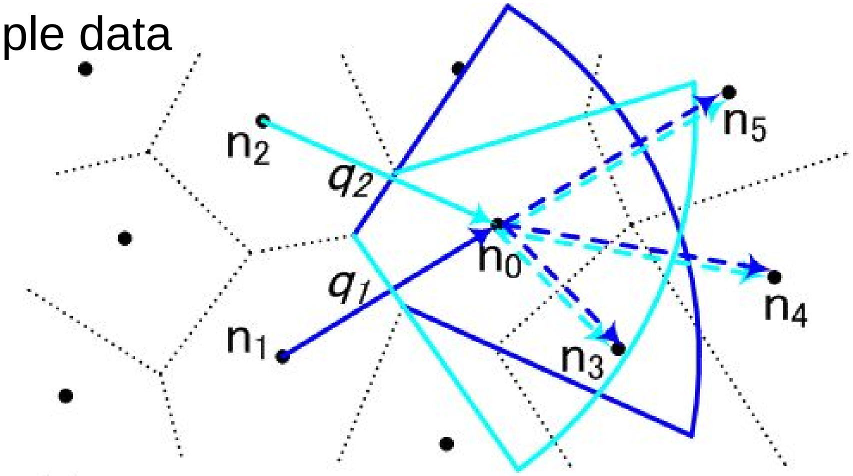
- **Processes for each node**

STEP 1. Generate **Send list** for classifying data to neighbors

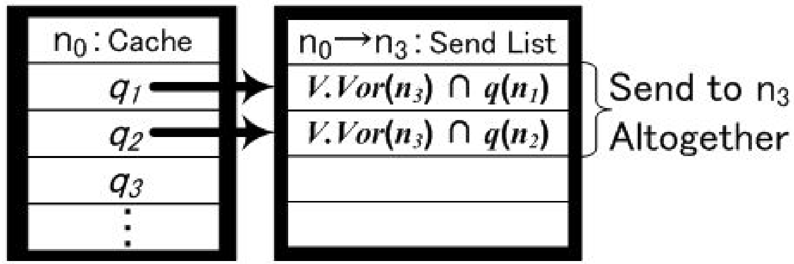
STEP 2. If $(V.Vor(n_i) \cap q(n_j))$
$$\text{sendList.add}(V.Vor(n_i) \cap q(n_j));$$

STEP 3. Combine multiple data and send at a particular time interval t

- Appropriate t as future works



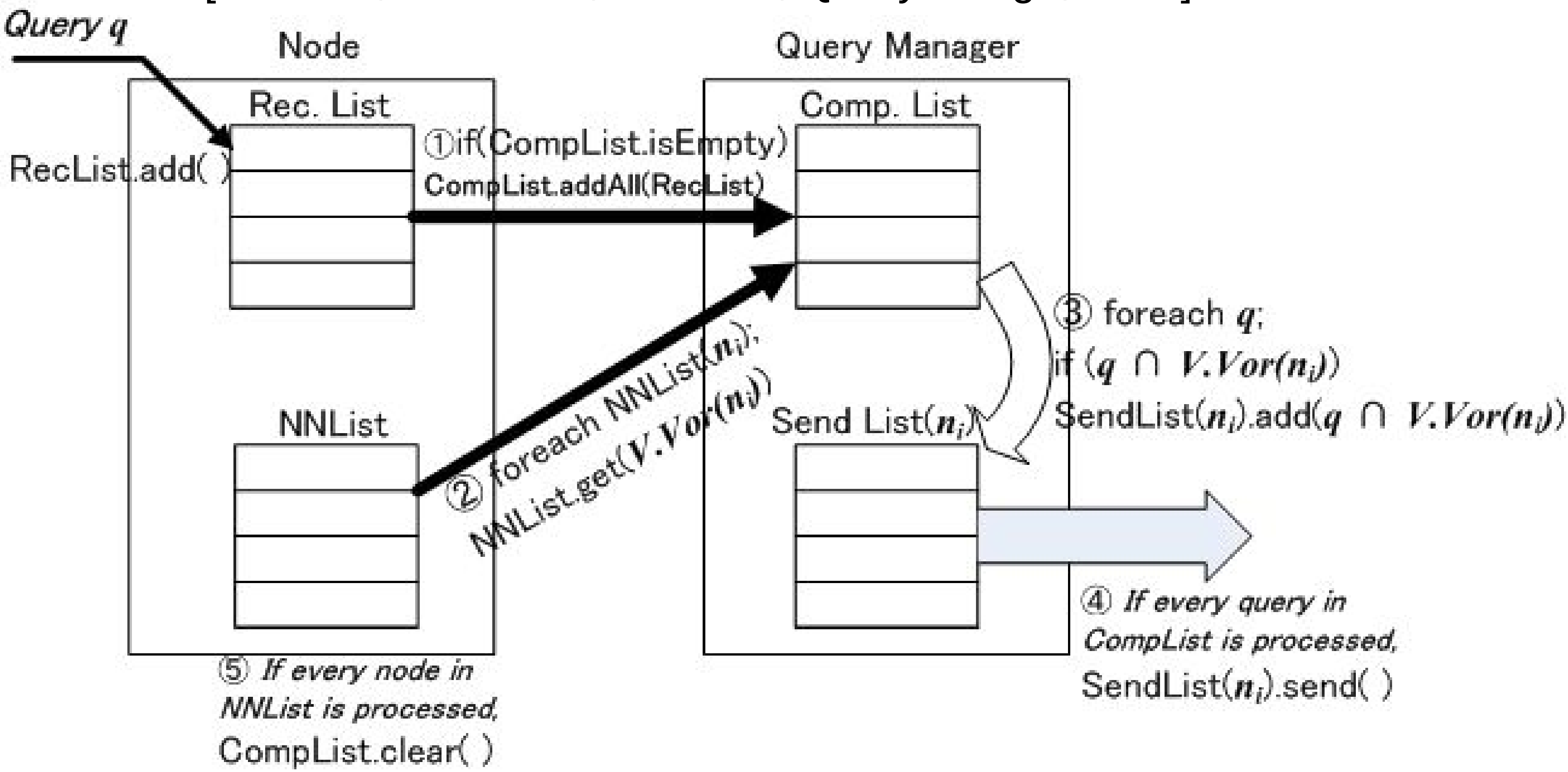
(a) Determination of Neighbors to Send Query



(b) Classification of Queries for Neighbor Node n_3

Data Structure

- Three lists: Receive List (Rec. List), Neighbor Node List (NN List), Computation List (Comp. List)
- Queries transferred between nodes
  - [Node ID, Packet ID, Location, Query Range, Time]



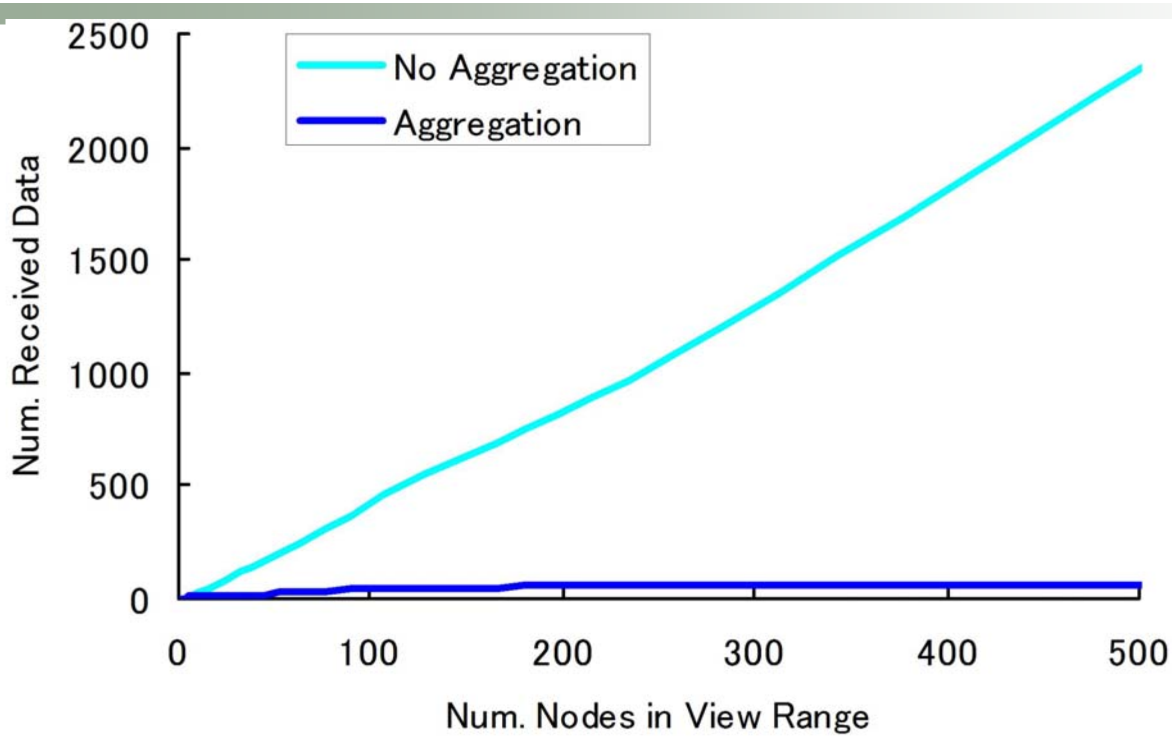
Evaluation

- Verify the efficiency of our method from node congestion within the view range
- Examine how the node density affects the CPU, network, and cache load with/without aggregation method

- **Simulation Settings**
  - **View Range**: Circular range
    - Size and shape equal for every node
  - **Node Distribution**: Nodes from 2 to 500 within view range
    - **Few nodes (low density)** to **many nodes (high density)**
  - **Data Transfer**: Nodes send data per each step
    - Nodes and network assumed to have equal performances

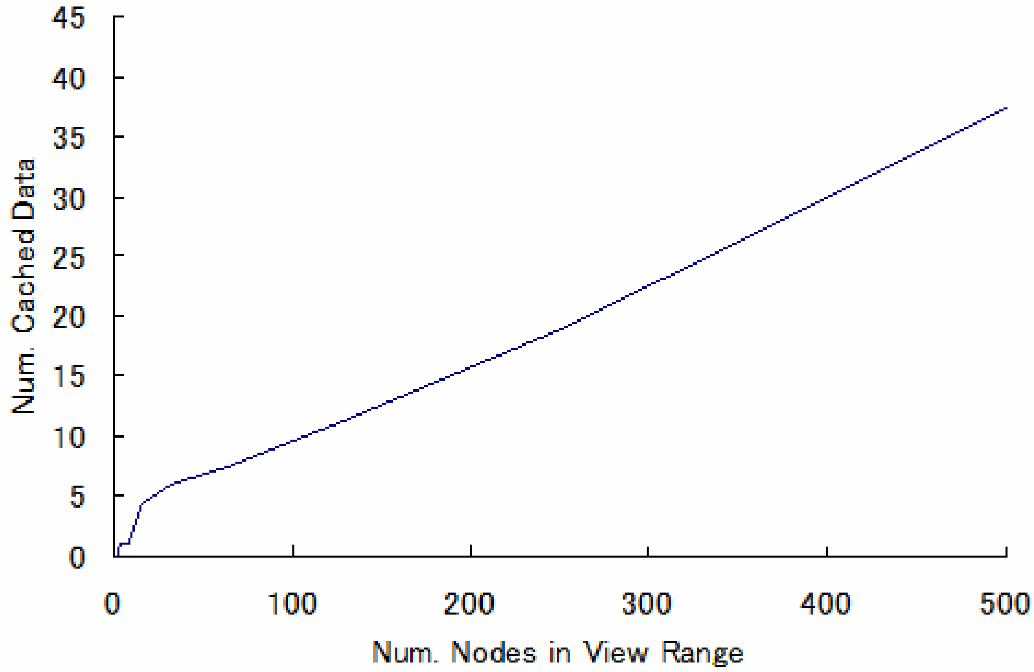
- **Methods**
  - **Aggregation**: Package multiple data and send to neighbor nodes
  - **No Aggregation**: Just cast data to neighbor nodes

CPU Load per Node



- **Evaluation method**
  - x axis: Number of nodes within the view range
  - y axis: Number of the entire received data of a root node
- **Evaluation results**
  - CPU load is significantly reduced with aggregation method
  - As the total data transferred between nodes are low, network load can be reduced as well

Average Cached Data per Node



- **Evaluation method**
  - x axis: Number of nodes within the view range
  - y axis: Average Number of cached data for a root node
- **Evaluation results**
  - No aggregation method: Just casts the received data, and thus does not cache data
  - Aggregation method: Number of cached data rises proportionally according to number of nodes

Discussions

- Simulations performed with step-by-step data transfer
- Results of analyses
  - **Node and network load significantly** reduced
  - **Number of caches increase moderately** according to node increase
- Following points for implementation in real environment:
  - **Data transfer speed** affected by CPU performance and network congestion
  - **Number of receivable data** should be controlled according to data volume and number of cached data
  - **Minimum data transfer latency** required for interactions and construction of view
- Important to determine an **appropriate interval time** for caching data

Related Works

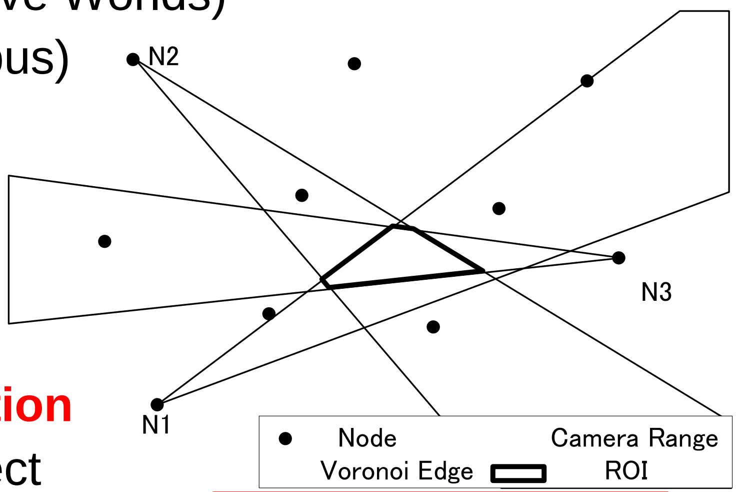
- **Network Scalability**
  - S.-Y. Hu, *P2P-NVE*, 2007
    - Forwarding model for disseminating data to surrounding nodes
  - Y.Kawahara, *IEEE ICCS*, 2002
    - Adjustments of direct connection and multihop communication nodes considering their distance
- **System Scalability**
  - S. Rueda, *IEEE VR*, 2007
    - System throughput and response time according to the types of node distribution
- **Nodes Interaction**
  - B. Knutsson, *IEEE Comp and Comm Societies*, 2004
    - Divisions of regions, and multicast tree for node communications within the same region

- Our focus: Node density problems within the view range
  - CPU and network load reduction for congested nodes within view range
  - Remote access to distant nodes within view range

Application Fields

- For **collecting local data** from the surrounding nodes on P2P network

- Application Examples
  - GUI construction for **Virtual Collaborative Space**
    - Massive Multiplayer Online Games (Everquest, Ultima Online)
    - Social Virtual Worlds (Second Life, Active Worlds)
    - Educational Environment (Digital Campus)
  - **Sensor networks** for aggregation of surrounding data
    - Target object extraction gazed by many people simultaneously
    - Necessary to reduce **network congestion** at a surrounding node set of target object



ROI: Region of Interest （Region w/ target object)

R. Nishide, *P2P-NVE*, 2007

Conclusion

- Data aggregation tree on SDN
  - **Achieve both advantages** of SDN and aggregation tree, essential for visualization and interaction in space
    - SDN: Geocast on SDN to **avoid data transfer delay**
    - Aggregation Tree: Caching scheme to **reduce CPU and network load**
- Numerical Simulation
  - Our method works efficiently in terms of node/network load distribution
  - Number of data in a cache rises moderately according to the increase of nodes
- Future Works
  - Acquire appropriate interval time lengths for caching data
  - Examine the amount of data loss due to packets transfer frequency
  - Verification of our method in real environments

Thank you for your kind attention