A Node Replication Method to Guarantee Reachability for P2P Sensor Data Stream Delivery System on Heterogeneous Churn Situations

Tomoya Kawakami$^{1, 2}$, Yoshimasa Ishi$^2$, Tomoki Yoshihisa$^2$, Yuuichi Teranishi$^{3, 2}$

$^1$ Nara Institute of Science and Technology, Japan
$^2$ Osaka University, Japan
$^3$ National Institute of Information and Communications Technology, Japan

2015/07/05
Tunghai University, Taichung, Taiwan
Background (1/2)

- Sensor data stream delivery infrastructure on Cloud
  - Huge number of streams must be accommodated in the future IoT/M2M
  - P2P-based stream data delivery is an approach to reduce the load concentration by scale out keeping resilience against churn of computation resources

- Consumes computation resources for stream delivery and user access
- Accepts data sources/data users as many as possible
Background (1/2)

- Sensor data stream delivery infrastructure on Cloud
  - Huge number of streams must be accommodated in the future IoT/M2M
  - P2P-based stream data delivery is approach to reduce the load concentration by scale out keeping resilience against churn of computation resources

Stream data delivery infrastructure on Cloud

- Stream data sources: Environment Sensor, Camera sensor
- Access Servers
- Stream data users

P2P-based stream data delivery

Load is distributed
Requirements for the stream differ by types of applications/specs of the destinations

- Spatial resolutions (e.g. picture quality of camera sensors)
- Temporal resolutions (collection cycles of the sensor data)
Sensor Data Stream Delivery to Accommodate Heterogeneous Cycles

- We have proposed a method to accommodate multiple cycles and construct delivery paths based on the cycles
  - In order to handle a huge number of streams and destinations, processing computers (nodes) relay the data

Sensor data stream $S_i$ - Sent from sources

Overlay network

Node $N_i$
- Communicate each other
- Relay data based on cycles or times

Destination $D_i$
- Request by streams and cycles
- Receive data only

Assign streams, cycles, times or destinations to balance the loads of each node

- Selectable cycles are specified and the timetable whose length is the lowest common multiple is generated
- In order to construct by each node individually, distributed hashing is used to assign

We propose a technique to guarantee a specific reachability to destinations even if nodes churn on heterogeneous situations
Framework to Construct Delivery Paths

Ring topology (e.g. Chord [Stoica et.al 2003]) for ID Space

\[ ID_{N_1} = h(x) \]

Hash function (e.g. MD5)

\( x: IP+port \text{ etc.} \)

The ring topology is shared for all stream sources
Framework to Construct Delivery Paths

Available cycles:
- 3 sec
- 2 sec
- 1 sec

Stream source A

<table>
<thead>
<tr>
<th>Periodic Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>..</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
<tr>
<td>2 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
<tr>
<td>3 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
</tbody>
</table>

Assigned to N₁

\(h'(stream=A, \ cycle=1, \ time=1)\)

\(h':\text{(stream data specification)}\)

\(h':\text{depends on the algorithm}\)
Framework to Construct Delivery Paths

Stream source A

Available cycles:
3 sec  2 sec  1 sec

<table>
<thead>
<tr>
<th>Periodic Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>..</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle</td>
<td>1 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
<tr>
<td></td>
<td>2 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
<tr>
<td></td>
<td>3 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
</tbody>
</table>

Timetable

Available cycles:
3 sec  2 sec  1 sec
Framework to Construct Delivery Paths

Stream source A

Available cycles:
3 sec  2 sec  1 sec

<table>
<thead>
<tr>
<th>Periodic Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>..</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle</td>
<td>1 sec</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>..</td>
</tr>
<tr>
<td></td>
<td>2 sec</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>..</td>
</tr>
<tr>
<td></td>
<td>3 sec</td>
<td>o</td>
<td></td>
<td>o</td>
<td></td>
<td>o</td>
<td>..</td>
</tr>
</tbody>
</table>

Diagram showing delivery paths with nodes labeled N1 to N6.
Framework to Construct Delivery Paths

Stream source A

Available cycles:
3 sec  2 sec  1 sec

Calculate and construct delivery path on each node by ring-search (such as Chord) using the stream data specification (depends on the algorithm)
Framework to Construct Delivery Paths

![Diagram showing network nodes and collection intervals]

- **Stream source**
- **Stream destination**
- **Collection interval:** 3 sec
- **Ring-search** the corresponding node for required collection interval

<table>
<thead>
<tr>
<th>Periodic Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>..</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle 1 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
<tr>
<td>Cycle 2 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
<td></td>
</tr>
<tr>
<td>Cycle 3 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timetable
Framework to Construct Delivery Paths

Stream source

Stream destination

Collection interval: 3 sec

Timetable

<table>
<thead>
<tr>
<th>Periodic Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>..</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 sec</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>..</td>
</tr>
<tr>
<td>2 sec</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td></td>
<td>..</td>
</tr>
<tr>
<td>3 sec</td>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td>..</td>
</tr>
</tbody>
</table>

Cycle

1 sec 2 sec 3 sec
Framework to Construct Delivery Paths

Collection interval: 3 sec

starts streaming!
Node Assignment Algorithm

- Based on the LCF algorithm [Kawakami et.al. 2012], in which delivery path can be calculated by the cycle variation
- Assumes uniform distribution of requests for each cycle
- Nodes are grouped by cycles for each stream

Sample node assignment (8 nodes, cycle=1,2,3)
Assignment of Cycle Group

- Divide the hash space based on the rate of data for destinations
- The shorter cycle groups have more assigned nodes

Ex.) 8 nodes, selectable cycle 1, 2, 3 (LCM 6)

1/1 : 1/2 : 1/3 = 6 : 3 : 2

The less divisor times have less assigned nodes, and the node failure effects to more nodes and destinations

Paths among groups is based on our previous methods called LCF
Successor List

- Used for ring-based DHT (Distribute Hash Table) such as Chord
- If the length is $k$, each node has links to $k$-next nodes (successors)

Keep high redundancy by sending same data to successors
Redundancy by Data Replication

- Each node also sends data to his successors (replication nodes)
- Also destinations determine the relayed nodes including replication nodes

Communication loads among nodes are increased, but the influence of node failure becomes small.
Algorithm to Determine the Number of Replication Nodes

- We determine the number of replication nodes based on assuming reliability of service (system), roughly expected reliability of nodes, and so on.
- Each node makes $v$ replication nodes at each time satisfying the following formulas:

Non-longest cycle group:

$$1 - \prod_{u=0}^{v} (1 - R_u) \geq \sqrt{SR_t}$$

Longest cycle group:

$$\left(1 - \prod_{u=0}^{v} (1 - R_u)\right) \frac{M_t + E_t}{\sum_{i=1}^{c} m_{it}} \geq SR_t$$

$SR_t$: Assuming system reliability (reachability to destinations) at time $t$ ($0 \leq SR_t \leq 1$)
$R_u$: Reliability of candidate nodes as an index of churn rate ($0 \leq R_u \leq 1$, $R_0$ : assigned node)
$m_{it}$: Amount of data in the non-longest cycle group $i$ ($c$ selectable groups)
$M_i$: Amount of data in the longest cycle group

**Ex.) $N_1$:** $t = 0$, Cycle 1 group (non-longest cycle group on $t$)
- Reliability of nodes: **0.8** ($N_1$, assigned node), 0.7 ($N_2$), 0.6 ($N_3$), 0.5 ($N_4$)
- $SR_t = 0.9$

Case of $v = 2$

$$1 - (1 - 0.8)(1 - 0.7)(1 - 0.6) = 0.976 \geq \sqrt{0.9} \Rightarrow N_1 \text{ makes } N_2 \text{ and } N_3 \text{ replication nodes at } t$$
Evaluation by Simulation

- Nodes are placed to not be gathered extremely on ring topology
- Specified system reliability: 0.9 (e.g., the reachability to destinations is 90%)
- Selectable delivery cycles: bet. 1 and 10 (10 variants, at random)
  - Least common multiple: 2520 ($0 \leq t \leq 2519$)
  - Each cycle has about 100 destinations
- Each method executes 10 times and calculates the avg. of results
  - 0, 1, 2, 4, 8: the num. of replication nodes is static for all times
  - Static: the num. of replication nodes is the least value to satisfy the specified system reliability and static for all times
  - Proposed: the num. of replication nodes is changed for each time
Min. of Inst. System Reliability
(Cycle: 1 - 10, $SR = 0.9, t = 0$)

Specified reliability 0.9 is satisfied appropriately

Constant scenario:
All nodes churn by the rate on the x-axis

Random scenario:
Each node churns by the rate bet. 0.0 and 1.0

![Graph showing Min. of Instantaneous System Reliability vs. Churn Rate with different scenarios and reliability levels.](image)
Total Num. of Replication Nodes
(Cycle: 1 - 10, \( SR = 0.9, t = 0 \))

Unnecessary replications are reduced for each time satisfying a specified reliability

Constant scenario:
All nodes churn by the rate on the x-axis

Random scenario:
Each node churns by the rate between 0.0 and 1.0
Conclusion

• A Node Replication Method to Guarantee Reachability for P2P Sensor Data Stream Delivery System on Heterogeneous Churn Situations
  – Processing computers (nodes) relay data to destinations based on cycles
  – Each node constructs delivery paths individually at each time by distributed hashing
  – Redundancy of node assignment by a successor list
    • Length of each successor list is determined based on our proposed algorithm
    • We confirmed that our proposed algorithm determines suitable length of a successor list for various churn situations

• Future work
  – Solution for an environment where the numbers of destinations are unbalanced for each cycle