An Efficient Anycast Scheme for Discovering K Services in Mobile Ad-hoc Networks


Department of Information Engineering and Computer Science, Feng Chia University
10/27/2008
Agenda

- Introduction
- Anycast Scheme
  - Anycast Tree Establishment Scheme
  - Service Information Collection Scheme
  - Service Discovery Scheme
  - Maintenance Schemes
- Experimental Environment and Results
- Conclusions
Introduction (1/2)

- Anycast service
  - Provide a one-to-any bidirectional transmission model
- Mobile ad-hoc networks
  - Multi-hop communications
  - Highly dynamic topology
- Most anycast schemes presented in MANETs are used for discovering one service
- Many distributed applications must work with many servers simultaneously
  - RPC and NTP
  - Threshold cryptography: ITTC project, COCA, and MOCA
Introduction (2/2)

- There are some extended anycast schemes that can support the discovery of $k$ services
  - Manycast
  - K-anycast

- However, when the senders or receivers increase
  - Low satisfactory ratio, high control overhead, and high searching latency

- Do not have enough service information
  - Locate the service providers

- Lacks a hierarchical structure
  - Reduce the service discovery overhead
AnyKast Scheme

- In this work, we propose an efficient anycast scheme (AnyKast)
- Anycast tree establishment scheme
  - Avoid unnecessary message transmission
- Service information collection scheme
  - Collect service information for selecting k services
- Service discovery scheme
  - Use service information to search k services
  - Enhance the reliability of our proposed scheme
- Anycast tree and service information maintenance
  - Promote service information accuracy
  - Decrease the control overhead
Anycast Tree Establishment Scheme

Service Provider
ACH (anycast clusterhead)
CH (Clusterhead)

Prune Message
Update ServiceInfo to SIT Scope:2-1=1>0 \rightarrow \text{Forward}

Update ServiceInfo to SIT Scope:1-1=1 \rightarrow \text{Stop forwarding}

Update ServiceInfo to SIT
Store it's ServiceInfo + SREG\{ServiceInfo\} to SREG\{ServiceInfo\}
Service Discovery Scheme

Service Selection

<table>
<thead>
<tr>
<th>N_i</th>
<th>ID_i</th>
<th>T_i</th>
<th>S_{i,j}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>74</td>
<td>9</td>
<td>1 0 1 0 5 0 0 0 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
<td>15</td>
<td>0 4 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1 0 0 6 4 1 2 0 0 2 4</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>10</td>
<td>0 0 3 3 9 5 1 3 1 2 3</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>12</td>
<td>0 0 0 1 0 0 1 4 2 0 3</td>
</tr>
</tbody>
</table>
Anycast Tree Maintenance Scheme
Service Information Maintenance

- Service information piggybacking
  - When an ACH or a SSN forwards SREQ and SREP messages, the service information will be piggybacked

- Periodical inquiry
  - When Inquiry Timer is timeout
    - A SSN will send INQI message to nodes whose service information is unknown or stale
  - When a SSN or a leaf node receives INQI message
    - Create REPI including its service information and send to sender
Experimental Environment for the AnyKast Scheme

- Simulator: GloMoSim
- Nodes: 250
- Area: 1500m × 1500m
- Tx_range: 250m
- Mobility model: Random waypoint
  - Max Speed: 2m/s
- Number of senders: 1 ~ 10
- Number of services: 25 ~ 125
- Table timer: 20s
- Inquiry timer: 25s
Experimental Results of the AnyKast Scheme (1/4)

- Varying Number of Request Services (1/2)
Experimental Results of the AnyKast Scheme (2/4)

- Varying Number of Request Services (2/2)
Experimental Results of the AnyKast Scheme (3/4)

- Varying Number of Senders (1/2)
Experimental Results of the AnyKast Scheme (4/4)

- Varying Number of Senders (2/2)
Conclusions

- This work presented an efficient anycast scheme for discovering \( k \) services in MANETs.
- The established anycast tree can decrease the cost of unnecessary message transmission.
- Service discovery scheme use the service information to effectively search \( k \) services.
- Our maintenance schemes can reduce the control overhead and enhance service information correctness.
- The simulation results demonstrate that our AnyKast scheme can effectively discover \( k \) services and lower control overhead.
Appendix
## Comparisons of Various Schemes

<table>
<thead>
<tr>
<th></th>
<th>Hierarchical structure</th>
<th>Collect service information</th>
<th>Discovery strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnyKast</td>
<td>Yes (clustering + virtual backbone)</td>
<td>Yes (SSN)</td>
<td>Scope flooding + Service selection</td>
</tr>
<tr>
<td>VBAF</td>
<td>Yes (clustering + virtual backbone)</td>
<td>No</td>
<td>Scope flooding + Forwarding gate</td>
</tr>
<tr>
<td>AT-Flooding</td>
<td>Yes (clustering + virtual backbone)</td>
<td>No</td>
<td>Scope flooding</td>
</tr>
<tr>
<td>Manycast</td>
<td>No</td>
<td>No</td>
<td>Scope flooding</td>
</tr>
<tr>
<td>K-anycast</td>
<td>Yes (simple clustering)</td>
<td>Yes (CH)</td>
<td>Flooding</td>
</tr>
</tbody>
</table>
Experimental Results of the AnyKast Scheme (3/6)

- Reliability (1/2)

![Graph depicting reliability and K values for AnyKast, k-anycast, Manycast, and variations of AnyKast with different K']
Experimental Results of the AnyKast Scheme (4/6)

- Reliability (2/2)
Varying Flooding Scope in Static Ad-hoc Networks

![Graph showing varying flooding scope](image-url)
Varying Flooding Scope in Static Ad-hoc Networks
Varying Number of Request Services in Static Ad-hoc Networks

![Graph showing the satisfactory ratio of different services as K increases]

- AnyKast
- VBAF
- k-anycast
- AT-Flooding
- Manycast

The graph plots the satisfactory ratio against K, where K represents the number of request services. The satisfactory ratio decreases as K increases for all services, indicating a trend of decreasing satisfaction with more services.
Varying Number of Request Services in Static Ad-hoc Networks
Varying Number of Request Services in Static Ad-hoc Networks

Normalized Overhead vs. K

- AnyKast
- VBAF
- k-anycast
- AT-Flooding
- Manycast
Varying Number of Request Services in Static Ad-hoc Networks

![Graph showing control overhead for different protocols]

- AnyKast
- VBAF
- k-anycast
- AT-Flooding
- Manycast

Control Overhead:
- Reply Message
- Request Message
- Inquiry Message
- Register Message
Reliability in Static Ad-hoc Networks

The graph shows the satisfactory ratio for different network configurations as a function of the parameter $K$. The configurations include:

- k-anycast
- Manycast
- AnyKast
- AnyKast, $K' = 110\%K$
- AnyKast, $K' = 120\%K$
- AnyKast, $K' = 130\%K$
- AnyKast, $K' = 140\%K$
- AnyKast, $K' = 150\%K$

The x-axis represents the value of $K$, ranging from 5 to 50, while the y-axis shows the satisfactory ratio, ranging from 0.8 to 1.0.
Reliability in Static Ad-hoc Networks

The diagram shows the normalized overhead as a function of $K$, with different lines representing different schemes:

- k-anycast
- Manycast
- AnyKast
- AnyKast, $K'=110\%K$
- AnyKast, $K'=120\%K$
- AnyKast, $K'=130\%K$
- AnyKast, $K'=140\%K$
- AnyKast, $K'=150\%K$

The x-axis represents $K$ values ranging from 5 to 50, and the y-axis represents the normalized overhead, which increases as $K$ increases.
Varying Number of Senders in Static Ad-hoc Networks

![Graph showing varying number of senders in static ad-hoc networks. The graph plots the satisfactory ratio against the number of senders, with different line styles and markers for different protocols: AnyKast, VBAF, k-anycast, AT-Flooding, and Manycast.]
Varying Number of Senders in Static Ad-hoc Networks
Varying Flooding Scope in Mobile Ad-hoc Networks

![Graph showing varying flooding scope in mobile ad-hoc networks.](image-url)
Varying Flooding Scope in Mobile Ad-hoc Networks

![Graph showing varying flooding scope](image)
Varying Number of Request Services in Mobile Ad-hoc Networks
Varying Number of Request Services in Mobile Ad-hoc Networks

![Bar chart showing control overhead for different services: AnyKast, VBAF, k-anycast, AT-Flooding, Manycast. The chart compares reply message, request message, inquiry message, and register message control overheads.]
Anycast Tree Establishment Scheme (1/2)

- When the virtual backbone is established in MANETs, the leaf nodes on the virtual backbone can be identified.
- When $\text{CH} \notin \text{ACH}$ and $\text{Link}(\text{CH}) = 1$
  - CH will send a $\text{PRUNE}$ message to its parent node.
  - Set its $\text{BE}_\text{PRUNE}$ flag to TRUE, and record its parent ID.
Anycast Tree Establishment Scheme (2/2)

- When CH receives the *PRUNE* message
  - it will record that the state of the outgoing link is pruned
  - if $CH \notin ACH$ and $\text{Link}(CH) = 1$ (except the pruned link)
    - it will forward a *PRUNE* message to its parent node
Service Information Collection Scheme

- Collect service information as the selection metrics
- When a node becomes an ACH
  - Create a SREG to disseminate its information actively
- Scope flooding
  - The definition of scope
    - The number of SSNs
  - Scope↑
    - Accuracy of service information↑
    - Control overhead↑
### Service Information Collection Scheme (1/2)

- **Service Information Table (SIT)**
  - Space complexity: $O(nm)$

<table>
<thead>
<tr>
<th>$N_i$</th>
<th>Id of Neighbors ($ID_i$)</th>
<th>Lifetime ($T_i$)</th>
<th>Service Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>ID$_1$</td>
<td>$T_1$</td>
<td>$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ ... $S_{1,m}$</td>
</tr>
<tr>
<td>$N_2$</td>
<td>ID$_2$</td>
<td>$T_2$</td>
<td>$S_{2,1}$ $S_{2,2}$ $S_{2,3}$ ... $S_{2,m}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$N_n$</td>
<td>ID$_n$</td>
<td>$T_n$</td>
<td>$S_{n,1}$ $S_{n,2}$ $S_{n,3}$ ... $S_{n,m}$</td>
</tr>
</tbody>
</table>
Service Discovery Scheme

- When a sender node requests services
  - Create a SREQ message
  - In order to enhance the reliability,
    - sender can increase the number of requesting services $k$ to $k'$ ($k' > k$)
  - Set number of request services to $SREQ\{K_{No}\}$
- When a SSN receives a SREQ
  - To redistribute the number of request services
    - If total #services in SIT $< K_{No}$
    - If total #services in SIT $\geq K_{No}$
- When an ACH receives a SREQ message,
  - Create a SREP and send SREP message to sender node
  - Check $SREQ\{K_{No}\} - \text{NumService}$
    - If $SREQ\{K_{No}\} > 0$ :Send SREQ message
    - If $SREQ\{K_{No}\} \leq 0$:Stop forward
Anycast Tree Maintenance Scheme

- When an anycast service rejoins another CH
  - The CH will be an ACH
  - It will send an UNPRUNE message to its parent node and sets its BE_PRUNE flag to FALSE

- When CH receives the UNPRUNE message
  - If its BE_PRUNE flag is TRUE
    - CH will forward an UNPRUNE message to its parent node and set its BE_PRUNE flag to FALSE
  - If its BE_PRUNE flag is FALSE
    - CH will stop forwarding UNPRUNE message

- When an anycast service disjoins from ACH
  - If there are no other anycast services belonging to the ACH
    - The ACH will become a CH
  - If this CH is a leaf node
    - it will send a PRUNE message to its parent node and set its BE_PRUNE flag to TRUE.