UNIVERSITY COLLEGE LONDON

Department of Computer Science
MSc DCNDS

Use Case Specification

JMS Implementation for MANETs using Sociable nodes (JIMS)

Supervisor: Dr Cecilia Mascolo
Team members:
Haitian Chen
Liang Chen
Kavitha Gupta
Christos Savvidis
Wai Git Teo
Date: Monday, 6th September 2004

Use case specification

SU1. Start service in Pub/Sub model
Pre-conditions
1. The node has installed our system.
Flow of Events
1. Use **RU2**. But everything in RU2 has to now take the topic information into consideration.
Post-conditions
1. Beacon with topic information is sent.
2. A table of sociable nodes with topic information has established.
Exceptions

SU2. Send topical message
Pre-conditions
1. The table of sociable nodes has been established.
2. The message has been constructed.
Flow of Events
1. The publisher decides to send message to certain topic with certain TTL.
2. Use **RU3**. But everything in RU3 has to now take the topic information into consideration.
3. The publisher also sends the messages to the neighbour nodes who share the same interest. This can be done by selecting the nodes from the table whose "Time unit indicator" is 1 (i.e. means this node now is neighbouring with the publisher) and the Topic name is the same as the publisher's.
Post-conditions
1. The message has been sent and deleted.
Exceptions

SU3. Pull topical message
Pre-conditions
1. The table of sociable nodes has been established.
Flow of Events
1. The subscriber "subscribes" to certain topic by indicating his interests.
2. Use **RU4**. But everything in RU4 has to now take the topic information into consideration. (Other subscribers that share same interest may also be enquired.)
Post-conditions
1. The message has been pulled and notified.
Exceptions

SU4. Send topical messages in session
Pre-conditions
1. The table of sociable nodes has been established.
2. The messages have been constructed.
Flow of Events

1. The publisher decides to send messages to certain topic with certain TTL in certain session.
2. The system sends a message that indicates how many messages are there in this session.
3. Set maximum number of transmitted message (M) to -1.
4. While there is a new stable sociable node 4.1. While the number of messages sent to this node is smaller than the total number of messages in the session. 4.1.1. Use SU2. Send the message of the sequence number that is equal to (M+1)%numberOf Messages. 4.1.2. Increment M by 1.
Post-conditions
1. The messages have been sent and deleted.
2. The session is torn down.
Exceptions

SU5. Pull topical messages in session
Pre-conditions
1. The table of sociable nodes has been established.
Flow of Events
1. The subscriber "subscribes" to certain topic by indicating its interests in the beacon.
2. The subscriber receives a session set-up message indicating how many messages there are in this session, and what the time-out is.
3. While within the period of the time-out. 3.1. If there is a new message to receive, use **SU3**. But everything in SU3 has to now take the topic information into consideration. 3.2. Go back to 3.
4. If all messages have been received signalled by notifications. 4.1. Make all messages available to application level.
5. Else, 5.1. Delete all messages received. i.e. roll back.
Post-conditions
1. The messages in the session are either completely received or none is received.
Exceptions

Reused Use Cases

RU1. Anti-entropy message exchange	
Pre-conditions	
1. A link between two nodes has been set up.	
2. Two nodes involved are determined. E.g. in Pub/Sub model they must share the same interest; in P2P model they can be anything. But they are determined!	
3. The time the message should be kept in buffer before deleted is known. T4.	
Flow of Events	
1. The sender sends meta-data to the receiver.	
2. The receiver sends meta-data to the sender.	
3. While there are more messages to send	4. While there are more messages to receives.
5. While T4 hasn't expired 5.1. If a NACK is detected, resend the message	6. If a loss of message is detected. Send NACK.
7. The end of the use case.	
Post-conditions	
1. Two nodes carry the exactly same messages.	
Exceptions	

RU2. Start service in core model
Pre-conditions
1. The node has installed our system.
2. The threshold for sociability value is known. H1
3. The threshold for asociablility is known. H2
4. The time unit for counting different neighbour nodes is known. T1
5. The cycle of sending out sociable declaration beacon is known. T2
6. The cycle of table refreshing is known. T3
7. We simply assume nT1 = nT2 = T3.
8. The weight is known. W
9. The sociability formula is known.
10. All initial tables or values are zeros.
Flow of Events
1. The user starts the system.
2. The system starts listening to the network.
3. While t < T3 3.1. While t < T1 3.1.1. If a beacon is received 3.1.1.1.If the UNI detected does not exist in the table 3.1.1.1.1. Add an entry for this node with associated info sent by the beacon. 3.1.1.2.If the beacon declares the node as sociable. ***Extend*** <Sociable node detected>. Refer the extension case Discover sociable nodes. Continue. 3.2. (T1 has expired.) Count how many entries (i.e. different UNI) in the table, Ni. 3.3. Subtract Ni by Ni_{-1}, which is the entry count for the last time slot. D = Ni –

Ni-1. 3.4. Si ← α Si+(1−α)D （Si is the historic sociability, D is the number of different nodes met in the current time unit. α is chosen according to the importance of historic information） 3.5. Feed Si, and other context information (memory, power) to sociability formula to get the sociability value for current time unit. Ii 3.6. Check against the sociability threshold. 3.6.1. If Ii > = H1, a field in beacon is set declaring the node is sociable. 3.6.2. Else, the field is reset. 3.7. Send out the beacon.
4. T3 expires, update the table.
5. Go back to step 3.
Post-conditions
1. Beacons are sent.
2. Table for sociable node is established.
Exceptions

RU3. Send message
Pre-conditions
1. The table of sociable nodes has been established.
2. The threshold for associability is known. H2.
3. The TTL for the message is known.
4. The message has already been constructed.
Flow of Events
1. The system determines the waiting time derived from TTL.
2. While within the period of waiting time 2.1. The system picks out the candidate nodes referring to the threshold. 2.2. If there're new candidate sociable nodes available, the system will notify to send the message to them. Use **RU1**. 2.3. Else, continue
3. The message is deleted from the buffer.
Post-conditions
1. The message has been sent and deleted.
Exceptions

RU4. Pull message
Pre-conditions
1. The candidate sociable nodes have already been determined.
2. How often the node check out the sociable nodes for either topic-data or peer-data is known. T4, and T4 = T1.
Flow of Events
1. The receiver enquires sociable nodes listed in the table.
2. Use **RU1**.
3. The node notifies the successful of getting a new message.
4. T4 expires, go back to 1.
Post-conditions

1. The message has been received and notified.
Exceptions

RU5. Buffer management
Pre-conditions
1. N/A
Flow of Events
1. The node starts the buffer management service.
3. Get the current global time in the node, comparing with all the messages' expiration time (from timestamp and TTL) in the buffer. a. If there is any message times out, delete it
3. If the queue is full, update the status.
3. If a new message is received, 3.1 If the buffer is full when a new packet comes 3.11 Extract the timestamp, TTL and priority (P) from its head, and calculate how much lifetime (T) is left for this packet. Feed the P and T into a formula (get a combination value reflecting the lifetime and priority) 3.12 If the value is smaller than all the messages in the buffer, discard it. 3.13 Else, delete the packet with the smallest value in the buffer to make space for the new packet. 3.3 Go back to 3
Post-conditions
1. N/A
Exceptions

Extension Use Cases

EU1. Discover sociable nodes

Pre-conditions

1. This case extends reused use case “start service in core model”.
2. The same table used in the extended use case is available here too.

UNI	Associability	Time Unit Indicator
	(number of beacon received in T2)	1(stands for current time unit)
	(If the node is not sociable but normal node, value is -1. so they will fall to the bottom of the table and never get picked.)	0(stands for previous time unit)

This table is updated whenever a beacon is received.
This entire table is refreshed every T3.

Flow of Events

1. A sociable node has been detected as it declares itself.
2. The associability value (i.e. counter) is incremented by one.
3. The table is reordered according to the column of “Associability”.

Post-conditions

1. A table has been established. (When sending the message, draw the threshold across the table and pick out the candidate sociable nodes.)

Exception

EU2. Discover topical sociable nodes

Pre-conditions

1. This case extends use case “start service in Pub/Sub model”.

Flow of Events

5. Use **EU1**. But everything in EU1 has to now take the topic information into consideration.

Post-conditions

1. A table has been established with topic information.

Exception