Content & Context Aware Routing for Mobile WSNs

Gianpaolo Cugola
Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy
cugola@elet.polimi.it
http://home.dei.polimi.it/cugola

Outline

- An introduction to CBR & why CBR in WSNs
- On adding context-awareness to CBR
- CCBR: The API
- CCBR: Toward a protocol
  - The peculiarities of WASP scenarios
  - The fundamentals of a probabilistic protocol for CCBR in WSNs
- Conclusions and future work

Politecnico di Milano

What is CBR

- Conventional routing (Address-Based Routing)
  - The sender explicitly specifies the intended message recipients
  - Using a unicast or multicast address
- Content-Based Routing
  - Messages do not carry any (explicit) address...
  - ... they are (implicitly) addressed, based on their content
    - Nodes express their interests in receiving specific messages
    - The sender simply injects messages into the network
    - The network chooses the recipients based on the message content and on the expression of interests of each node

Politecnico di Milano

Why CBR (in general)

- CBR introduces a strong form of decoupling among communicating parties
  - Communication is asynchronous, multicast, anonymous, implicit
- Adding, removing or even moving components becomes trivial

Politecnico di Milano

Why CBR (in WSNs)

- WSNs interactions are mainly *data-centric*...
  - WSNs are designed to gather data and deliver it
- ... and *multicast*
  - Each sink collects data from different sensors
  - The data produced by each sensor may be of interest for different sinks
- Which routing for WSNs?
  - Mapping a multicast, data-centric interaction on top of a conventional (unicast, address-centric) routing protocol may be very inefficient
  - CBR perfectly fits a data-centric interaction

Politecnico di Milano

CBR & Context-Awareness

- Interaction in WSNs is often *context-aware*, e.g.
  - A farmer could be interested in having the temperature reading (each hour) of young cattle only
  - A doctor could be interested in being notified when something bad happens to patients accepted in her division, not in all patients of the entire hospital
  - A guest could be interested in searching for the nearest printer
- Encoding such context-awareness as part of message content in order to use standard CBR is possible...
- ... but can be inefficient

We need a context & content based routing protocol (CCBR)

The CCBR protocol: The API

listenFor(ComponentFilter, MessageFilter, AdditionalData, MessageListener, LeaseTime)

- Chooses the relevant components based on their current context
- Chooses the relevant messages based on their content
- Blindly transported to the relevant nodes
- A pointer to a function:
- An integer expressing the period of validity (in seconds) for the expression of interest

CCBR protocol: The API

setComponentProperties(Properties, DataListener)

- Advertizes the component's context
- Expressed as a set of tuples <attr. name, value>
- E.g., <age,12>

- A pointer to a function:
  void notifyData(AdditionalData)
  invoked when new data arrives (as part of a listenFor operation invoked somewhere in the network)

send(Message)

- Expressed as a set of tuples <attr. name, value>

Politecnico di Milano

Toward a protocol: The scenarios we consider

- The case of *mobile* WSNs is not rare
  - Nodes installed on animals…
  - … or people
- When mobility enters the picture things becomes much more complex
  - The topology of the network changes much more frequently than in traditional cases
- The few CBR protocols for WSNs do not explicitly consider the case of mobile nodes
  - When they do, they address delay-tolerant networking scenarios

Politecnico di Milano

Toward a protocol: The scenarios we consider

- Several WSN scenarios involve multiple "sinks"
  - Several sinks can be interested in acquiring data from the network
  - Some of them can be mobile
    - E.g., when the BAN of a person gathers data from the ambient WSN
    - E.g., when a doctor queries the BAN of her patients
  - There are scenarios in which every node (not only "sinks") is interested in receiving messages
    - E.g., a node may run the code for processing temperature readings coming from the set of nodes located in the same area to issue an alarm when the average exceeds a threshold
- The presence of multiple, possibly mobile "sinks" further complicates the scenario

Politecnico di Milano

Toward a protocol: General approach

- Use a probabilistic approach
  - To limit the traffic while keeping good delivery in presence of very frequent changes of topology
- Use link layer broadcast whenever possible
  - To minimize traffic
  - Taking advantage of an ad-hoc MAC capable of optimizing power usage for broadcast communication
- Introduce mechanisms capable of self adapting to different conditions
  - Level of mobility
  - Node density
  - Residual energy of nodes
  - Traffic patterns

Politecnico di Milano

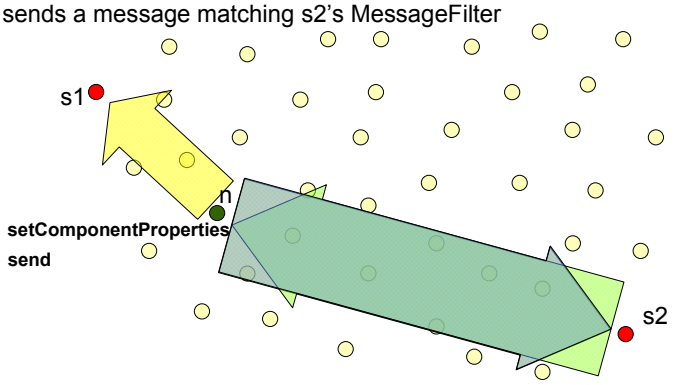
The CCBR protocol: Two possible solutions

- Nodes keep track of their distance from the "sinks" (i.e., those nodes that issued a listenFor)
- Solution 1
  - The component properties are routed toward the sinks...
  - ... where they are used to decide if and where to route the MessageFilters (by following back the route of matching properties)
  - Messages are sent only toward those sinks whose MessageFilter matches them
- Solution 2
  - Component properties are stored locally
  - MessageFilters flood the network (piggybacked on beacons used to update distances from the "sinks")...
  - ...and stored only by the nodes having the "right" component properties
  - Messages are sent only toward those sinks whose MessageFilter matches them

Politecnico di Milano

The CCBR protocol: First solution

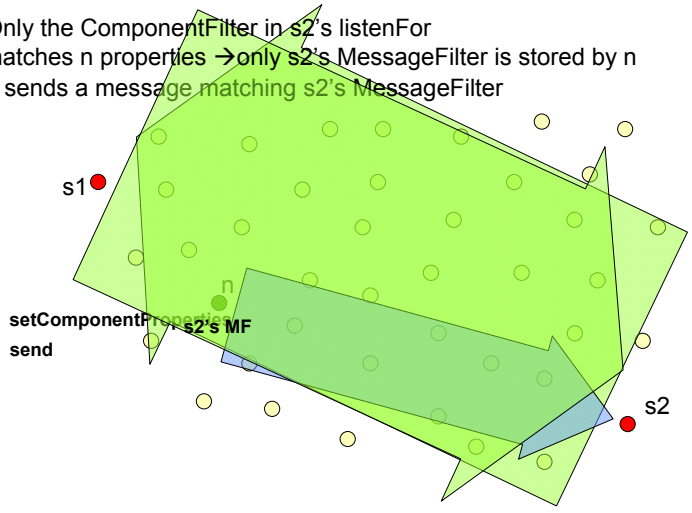
- s1 and s2 issued a listenFor
- Only the ComponentFilter in s2's listenFor matches n properties
  → only s2's MessageFilter is "sucked" by n's setComponentProperties
- n sends a message matching s2's MessageFilter



Politecnico di Milano

The CCBR protocol: Second solution

- s1 and s2 issue a listenFor
- Only the ComponentFilter in s2's listenFor matches n properties →only s2's MessageFilter is stored by n
- n sends a message matching s2's MessageFilter



Politecnico di Milano

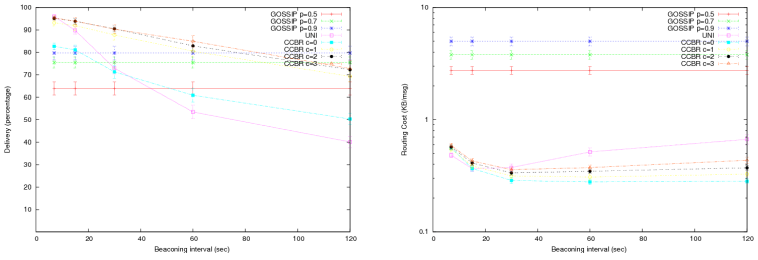
The CCBR protocol: Common part

- All the communication uses probabilistic, broadcast based mechanisms
  - Each node locally decides (probabilistically) if forwarding a packet (i.e., Message, Property, MessageFilter)
    - Using its estimate distance from the sinks and the distance of the previous forwarder (plus other info, e.g., remaining battery)
  - Forwarding is done in broadcast
    - Several nodes may hear the message
    - Excessive rebroadcast is limited by overhearing
      - Previous experience shows that this approach provides self-adaptation to network density
    - Overhearing is also used as an implicit ack
  - The level of mobility can be estimated (locally) and used to determine various parameters
    - The frequency of refresh for routing information
    - The threshold to re-forward a packet

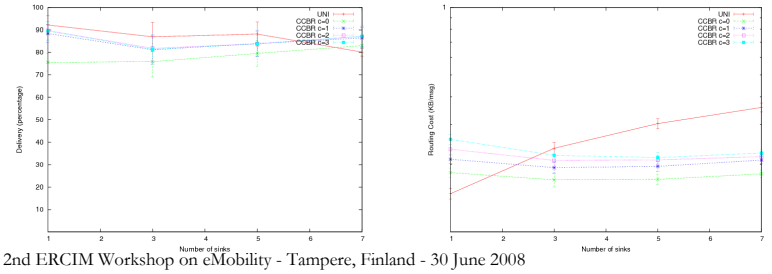
Politecnico di Milano

Some initial results (for the 2nd solution)

Default scenario: 50 sensors, 3 sinks, 0.5 Km^2, walking mobility, 0.1 msg/s, selectivity of filters: 10%



Fixed scenario: Same as above but no mobility. Growing number of sinks, all receiving messages



Conclusions & future work

- CBR is a promising approach for routing in WSNs
- CCBR extends CBR including context-aware information to further reduce traffic
- Previous experience and initial results in simulated scenarios make us confident that a probabilistic approach fits the peculiarities of a mobile, multi-sink scenario
- We are working on implementing CCBR on real nodes (TelosB using TinyOS and FreeRTOS)
- Open issues
  - Before the end of the WASP project we want to explore the possibility of adding "in network processing" capabilities to CCBR
    - e.g., to specify an "average" filter and let CCBR choose where to execute it
  - It could be also interesting to explore RT (and other QOS issues) in CCBR