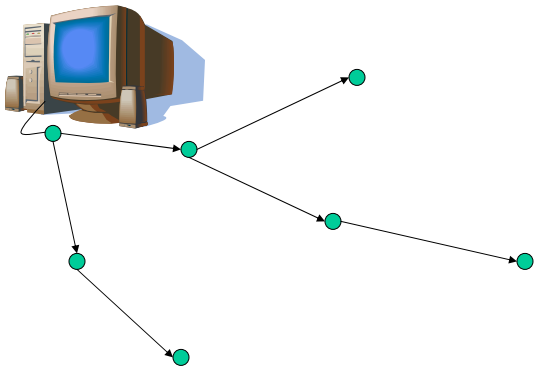
Stream: Low Overhead Wireless Reprogramming for Sensor Networks

Rajesh Krishna Panta
Issa Khalil
Saurabh Bagchi

Presentation at Infocom 2007

Introduction: What is sensor network reprogramming?



Motivation

- In-System Programming (ISP) is not always feasible
  - Large network
  - Sensor nodes may be situated at places which are difficult to reach
- Programming is not a one-time process
  - Requirements from the network or the environment in which the sensor nodes are deployed may change
  - May need to change certain parameters of the user application
  - May require to change the logic of the application itself (e.g. inherent unreliability in the wireless links may cause failures and require changing the application itself)

Requirements of Network Reprogramming Protocol

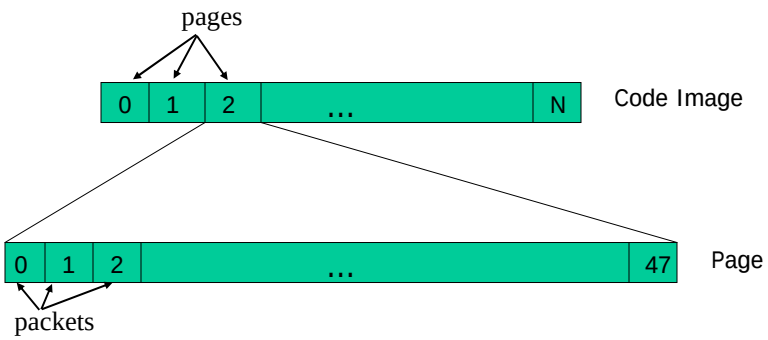
- All nodes in the network should receive the code completely
- Code upload should be fast so that sensor nodes can resume their normal function
- Minimize the resource cost of reprogramming – energy and bandwidth spent in disseminating code through the network

Network Reprogramming Protocols

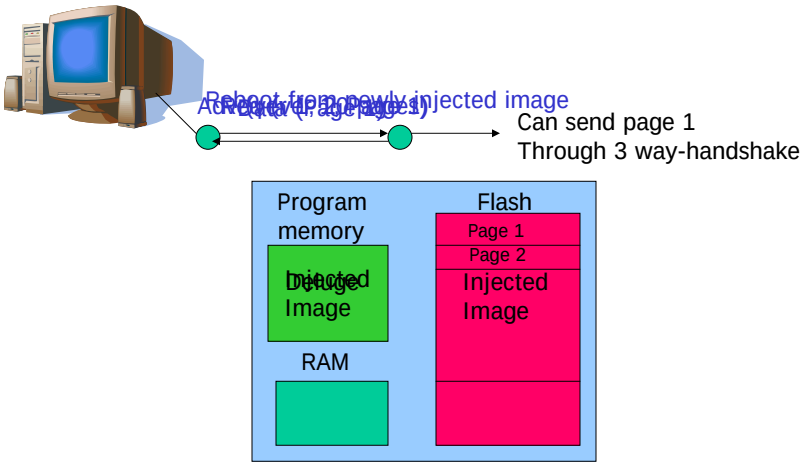
- First generation protocols
  - XNP: Single hop
  - MOAP: Multi hop; No pipelining of the segments of the code image
- State-of-the-art protocols
  - Deluge [Hui04]
    - Three way handshake: advertisement-request-data
    - Multi-hop
    - Pipelines segments of the code image
  - Freshet [Kras07]
    - Built on top of Deluge
    - Minimizes the energy consumption by putting the nodes to sleep during the time periods when they are not involved in reprogramming

[Hui04] J.W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale." SenSys 2004.
[Kras07] M.D. Krasniewski, R. Panta, S.Bagchi, C-L. Yang, W.J. Chappell,,"Energy-efficient, On-demand Reprogramming of Large-scale Sensor Networks," Under revision for ACM Trans. On Sensor Networks (TOSN).

Deluge: Structure of Code Image



Deluge: Code Dissemination



Problem with Existing Protocols (Deluge and Freshet)

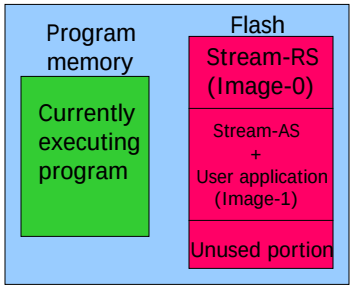
- They attach user application with the reprogramming component into one image
- Wirelessly transfer the entire image through the network
- Rationale for this
  - Enable the sensor nodes to receive future code updates
  - Sensor nodes are not multi-tasking
- Problems
  - High reprogramming time – network's functionality is degraded during the reprogramming period
  - Large number of bytes means more energy spent on reprogramming the network

Goals of Stream

- Reduce energy consumption during reprogramming: Through reducing the number of bytes transferred during reprogramming
- Minimize reprogramming time
- Portability across existing reprogramming protocols: Be independent of the core reprogramming protocol
- Handle incremental node deployment: Be able to reprogram new nodes that join the network

Stream's approach

- Stream segments the program image into Stream-RS (Stream Reprogramming Support) and Stream-AS (Stream Application Support)
- Stream-RS
  - Core reprogramming component
  - Preinstalled, before deployment, in all nodes
  - Stored in each node's Flash memory as image 0
- Stream-AS
  - A small subset of reprogramming component that is attached to the user application
  - Instead of wirelessly transferring through the network user application plus the entire reprogramming component, Stream transfers Stream-AS plus the user application
  - Stream-AS + user application is stored in each node's Flash as image 1



Protocol description: Getting ready for receiving code image

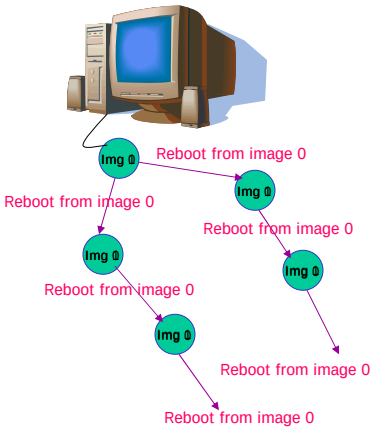
- Attach new user application to Stream-AS

```
components StreamASC;
Main.StdControl->StreamASC;
```

- Inject user application + Stream-AS as image 1 on the *base node*
- In response to the reboot command from the user, all nodes in the network reboot from image 0. This is accomplished as follows:
  - The base node executing image 1 initiates the process by generating a command to reboot from image 0. It broadcasts the reboot command to its one hop neighbors and itself reboots from image 0
  - When a node running the user application receives the reboot command, it rebroadcasts the reboot command and itself reboots from image 0

Stream-AS

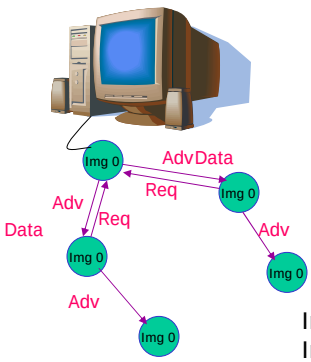
- Once reboot command reaches all nodes, they start running Stream-RS. Then the new application is injected into the network using Stream-RS



Img 0 : Reprogramming component
Img 1 : Modified user application

Protocol Description: Three way handshake

- Stream-RS reprograms the entire network. If built on Deluge, it does so by using the three way handshake method.



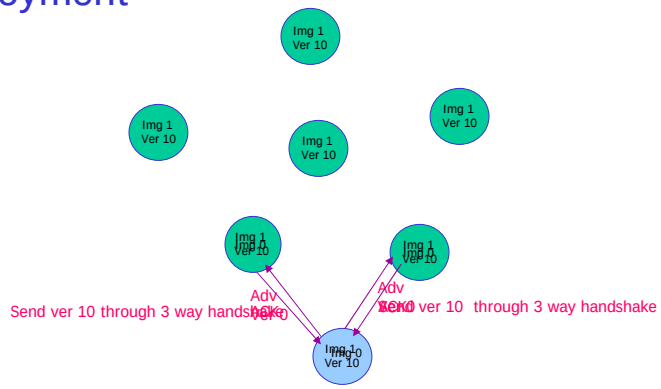
Img 0 : Reprogramming component
Img 1 : Modified user application

Protocol Description: Keeping all thy neighbors satisfied

- Each node maintains a set S containing the id's of the nodes from which it has received requests for the code
- Once the node downloads the new code completely, it performs a single-hop broadcast of an ACK indicating that it has completed downloading
- When a node n_1 receives the ACK from a node n_2, it removes the id of n_2 from its set S
- When the set S is empty and download is complete, the node reboots from image 1.
- Finally the entire network is reprogrammed and all nodes execute image 1 (Stream-AS + user application)

Handling incremental node deployments

- Pull based approach instead of traditional push based approach to handle incremental node deployment



Advantages of Stream: Reduction in Overhead Traffic

- The exact reduction in the number of bytes transmitted by Stream over Deluge depends upon the user application

User application	Size of User application	User application + Deluge	User application + Stream-AS
No radio communication (rare case)	1 page	21 pages	11 pages
Radio communication	11 pages	22 pages	12 pages

1 Page=48 Packets
1 Packet=36 Bytes

Other Advantages of Stream

- Reduction in reprogramming time and energy due to reduction in traffic that is uploaded through the network
- All nodes automatically reboot from user application after reprogramming is complete
- Less energy consumed by reprogramming component during *quiescent* phase
  - Sensor nodes do not spend their energy in continuous broadcasting of advertisements in quiescent phase
  - However, both new nodes joining the network and the new code pushed in by the base node are efficiently handled
- More efficient use of RAM and program memory

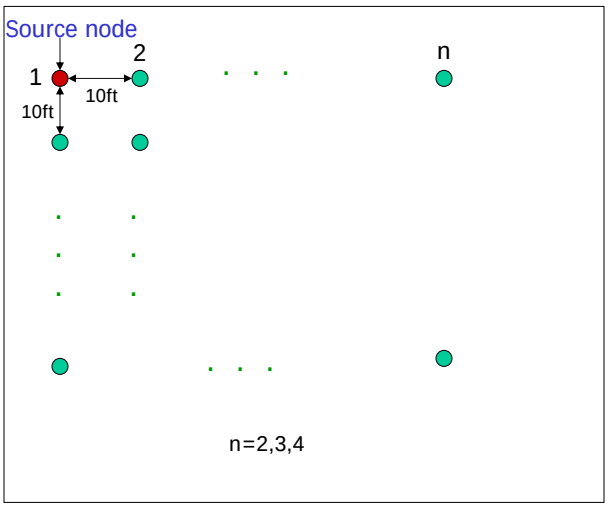
Implementation and Experiments

- We implemented Stream using nesC in TinyOS
- We performed test bed experiments using mica2 motes for various grid and linear topologies and TOSSIM simulations for grid topologies
- User application = 11 pages

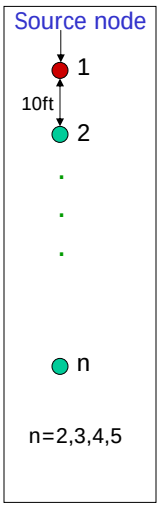
  User application + Stream-AS = 12 pages

  User application + Deluge = 22 pages
- Evaluation metrics
  - Network reprogramming time
  - Energy consumed during reprogramming (number of packets transmitted for reprogramming the network)

Test bed Experiments

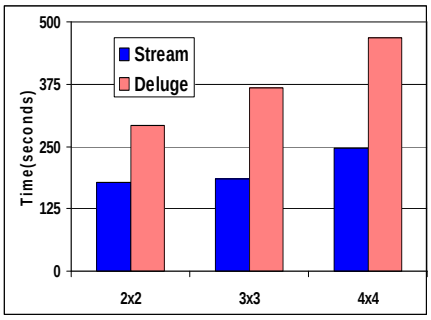


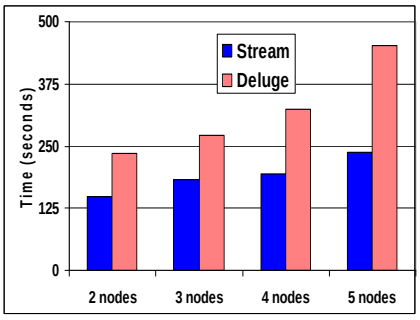
Grid topology



Linear topology

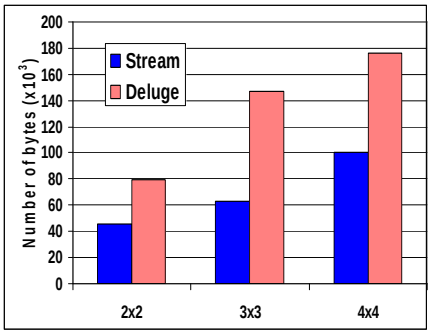
Test bed Results: Reprogramming Time

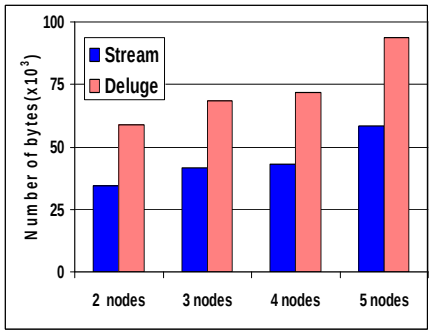




Up to 98% reduction in reprogramming time

Test bed Results: Network Traffic

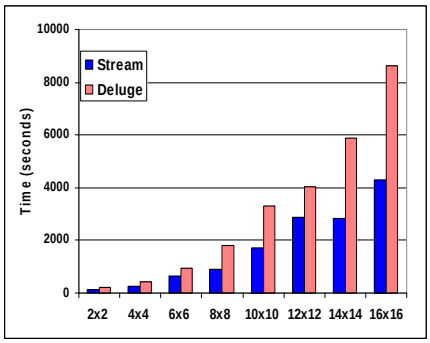


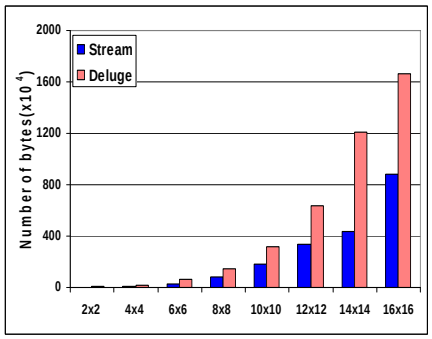


Up to 132% reduction in number of bytes transferred during reprogramming

Simulation Results: Network Size

- Grid Networks of various sizes with source situated at one corner of the grid
- Number of packets per page reduced from 48 to 24 packets

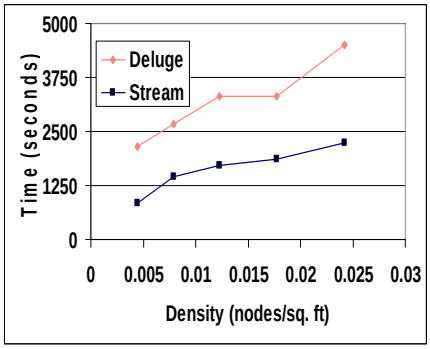


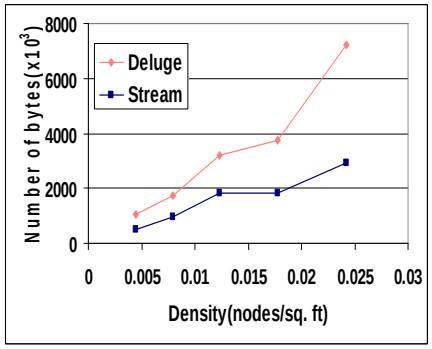


Up to 112% reduction in number of bytes transferred and 101% reduction in reprogramming time

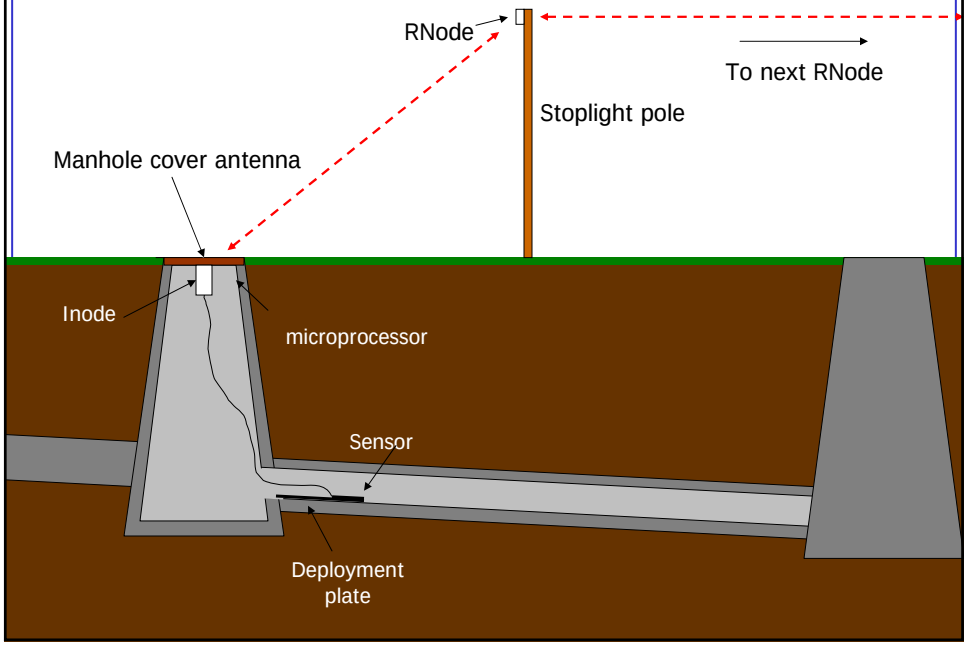
Simulation Results: Network Density

- We vary the number of nodes in a 90ft x 90ft area
- Nodes are arranged in grid fashion with uniform spacing between adjacent nodes

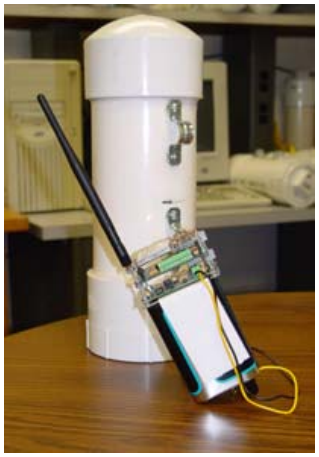




Emnet deployment



Emnet: Chasqui motes



Conclusion

- Stream significantly reduces the number of bytes transmitted over the wireless medium for reprogramming- decreases reprogramming time and energy
- Experiments conducted on test beds of mica2 nodes demonstrate up to 98% reduction in reprogramming time and 132% reduction in number of bytes transferred compared to Deluge. TOSSIM simulations show up to 101% reduction in reprogramming time and 112% reduction in number of bytes transferred compared to Deluge
- Stream can show improvement on any current reprogramming protocol since all send the entire reprogramming component with the user application

Take-away lessons and Future Work

- Two important aspects of sensor network reprogramming: reprogramming time and energy
  - Efficient interest based code dissemination in one hop: Solved problem (courtesy Deluge)
  - So we cut down the number of bytes transferred through the wireless medium
- Further work
  - Multiple originators
  - Heterogeneous network
  - Incremental reprogramming