Demo: Distributed Task Execution in Mobile Ad Hoc Networks using Attributed Task Graphs

Prithwish Basu   *Wang Ke*

Salma Abu Ayyash   *Thomas D.C. Little*

Department of Electrical and Computer Engineering, Boston University, Boston, MA.

{ pbasu, ke, saayyash, tdcl } @ bu.edu

Task Graphs (TGs) in MANETs

1. Pervasive networks must support distributed applications requiring exchange of information between multiple devices

2. MANETs inherently prone to network partitioning and link failures

→ **Resolution: Our Task Graph framework and execution environment**

Synopsis of the Work

•Demonstration of mapping of a distributed software application onto a MANET

•Decomposition of an application using Task Graphs to decouple its needs and structure from real devices

•Distribution of Task Graph onto the MANET

•Discussion of related work and extensions

How Does It Work?

•Map an application onto a logical Task Graph representation (*Modeling*)

•Map the Task Graph onto physical devices by finding the required resources (*Resource Discovery*).

•Protocols for a rapid and seamless response to disruptions in the network due to mobility and failures.

→ Our framework is *adaptive*, *resilient* and *flexible*.

Example Application: Wireless Polling or Sampling
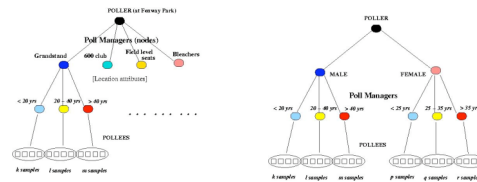
•Attribute based sampling in locations such as stadiums, crowded plazas, cars on highways

•Attributes may be location based or semantic

Embedding Task Graphs onto MANETs
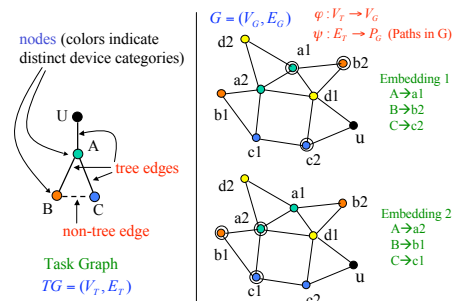
Proof-of-Concept Prototype on a MANET Testbed

•Heterogeneous computational elements: Notebook and handheld PCs with IEEE 802.11b cards

•Each device emulates a resource that can be queried

  –MANET routing protocol: OLSR

  –Mobility emulated using IP packet filtering (*iptables*)

  –Hughes Research Lab's MobiEmu tool dynamically inserts iptables rules in Linux kernel to change connectivity

•Task based Application mapping framework:

  –*taskd* (daemon for TG instantiation)

  –*taskapp* (user application)

Components of the Solution
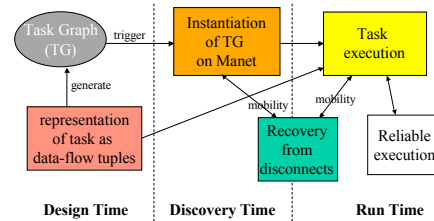
A Logical *Layered* View of the Task Execution Framework
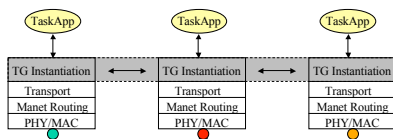
• Layering is logical – cross layer optimizations are possible

• TaskApp can be executed on remote devices

• Our focus/contribution: TG Instantiation/TaskApp layers

Application Framework

•*taskd*: handles instantiation and recovery from disconnections

  –User space implementation with Linux socket API

  –Single threaded daemon using `select()`

  –Uses UDP (with re-transmissions) and TCP

•*taskapp*: processes application data

  –To react to delays and failures in *taskd*

  –To buffer ADUs for reliable task execution

  –Can be already existing on a remote node

  –Or can be shipped as snippets of mobile code which can execute in a sandbox environment