MobilityFirst Network API use in Mobile Applications
Francesco Bronzino, Kiran Nagaraja, Dipankar Raychaudhuri, Ivan Seskar

Introduction to MobilityFirst

- Analysis of today's most used API (Berkeley Sockets):
  - Based on end-to-end communication
  - Tightly bounded to the TCP/IP services
  - Scarce support for mobility
  - Scarce support for data retrieval and service access
- What should a new API provide?
  - High flexibility for lower layers innovation and expansion
  - Provide full support for architecture services
  - Unbinding the API level from the transport layer
  - Choosing the right transport service from the user's intent
  - Abstracting the communication interface from the network architecture

A New Network API

- Intent of the user expressed through a URL
- GUID based
- Service IDs converted into a 16 bits flag used in the network routing
- Content centric operations available (i.e. get)

API operations
- open(profile-URL, src-GUID, profile-opts)
- send(message, dst-GUID, svc-flags)
- recv(buffer, [GUID-set])
- get(content-GUID, buffer, svc-opts)
- attach(GUID-set)
- close()

Client Stack Prototype

- JAVA (Android) and C (Linux) API prototype
- Usable with ARM based Android devices
- Multihoming capabilities for WiMAX enabled devices
- Distributed as a system library for easiness of deployment in Android SDK based applications
- Based on Hop protocol and use of pcap protocol to intercept packets

Challenge
- Host and network mobility
- Varying level of wireless connectivity
- Multi-homing

MobilityFirst approach
- Separation of naming and addressing: GNRS
- Hop-by-hop data transfer and storage aware routing
- Multi-homing support

MobilityFirst Architecture

Demo Scenario

Future work

- Refine the definition of the API and of the Service ID specifications.
- Perform a usability analysis of the API.
- Extend the client stack prototype by implementing new transport services.
- Implement new use cases scenarios to test the API effectiveness.