HAGGLE: Seamless Networking for Mobile Applications

Breno Jacinto
breno@gprt.ufpe.br

Introduction

- Users expect ubiquitous access to applications from their multiple devices
- Today
  - Applications are deeply dependent on the underlying networking architecture
  - Applications are forced to behave as the network demands
  - Example: Email and Web are totally dependent on DNS resolution

Introduction (2)

- Applications should adapt to different as network conditions changes
  - Email may be sent peer-to-peer if infrastructure is gone or if sender and destination are in proximity
  - Web content may be available from peer caches if the web server is offline or Internet connectivity is gone
  - Connectivity to the Internet should be possible by using intermediate nodes as a bridge

Motivation

- Alice and Bob are in a train
  - Alice wishes to forward Bob a discussion thread containing a document for review
  - However, given Internet connectivy limitations and costs (e.g, via GPRS), sending the email is difficult
  - Even if Alice and Bob were connected in a ad hoc fashion (i.e, 802.11 in ad hoc mode or Bluetooth), email does not work on these networks
  - Email is too tied to infrastructure: DNS, SMTP servers and POP servers

Motivation (2)

- Charlie wishes to read news during the train trip
  - Same Internet access problems as Alice and Bob
  - But, since reading the news is a popular activity, it may be possible that someone in the train has the news cached in their web browsers
  - Users could act as web caches for others around them, but they are not aware of this and even if they were, this wouldn't be trivial in current network architectures

The Haggle Approach

- Data transportation should not be a concern to the application, but exclusively for the networking architecture
- By separating concerns, Haggle provides a data-centric network architecture that internally manages the task of handling and propagating data
- This way applications adapt automatically to changing networking conditions

The Haggle Approach(2)

- Considers the use of multiple networking technologies at the same time
- Use three-level late-binding mechanism
  - **Interfaces:** For choosing the proper interface to use, considering a balance between interface characteristics and user and application preferences
  - **Protocols:**To support different routing protocols, by providing application adaptation in a given circumstance
  - **Names:** To support specification of services, individuals or devices

The Haggle Approach(3)

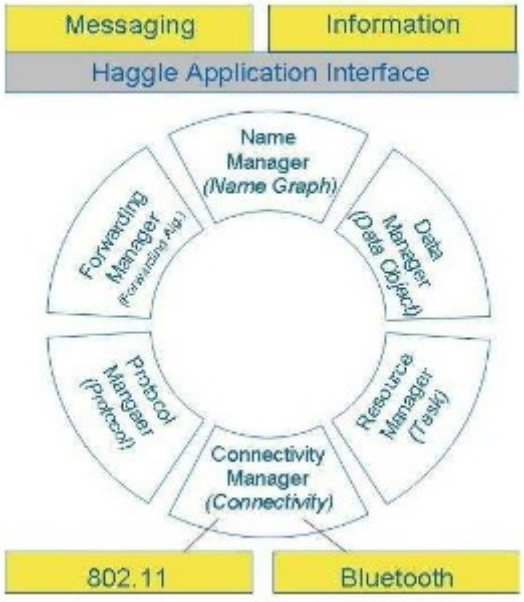
- Exposure of Persistent Data and Metadata
  - Goal: To provide support for data-driven networking
  - By exposing metadata for the browser's web cache, it's possible to answer queries for keyword-matching, for example
  - Two classes of metadata are provided: attribute tags and relationships
  - Data Objects (DOs) can be tagged with key/value pairs and relations can be established using directed edges

The Haggle Approach(4)

- Centralized Resource Management
  - All requests for manageable resources are issued as tasks to a centralized module which dictates which actions are allowed to proceed in accordance with current context and user-specified policies

The Haggle Architecture

- Eliminates layering above the data-link, supporting application-driven message forwarding, instead of delegating this responsibility to the network layer
- Six managers are organized in a layerless fashion
- All managers can communicate with each other via APIs



The Haggle Architecture (2)

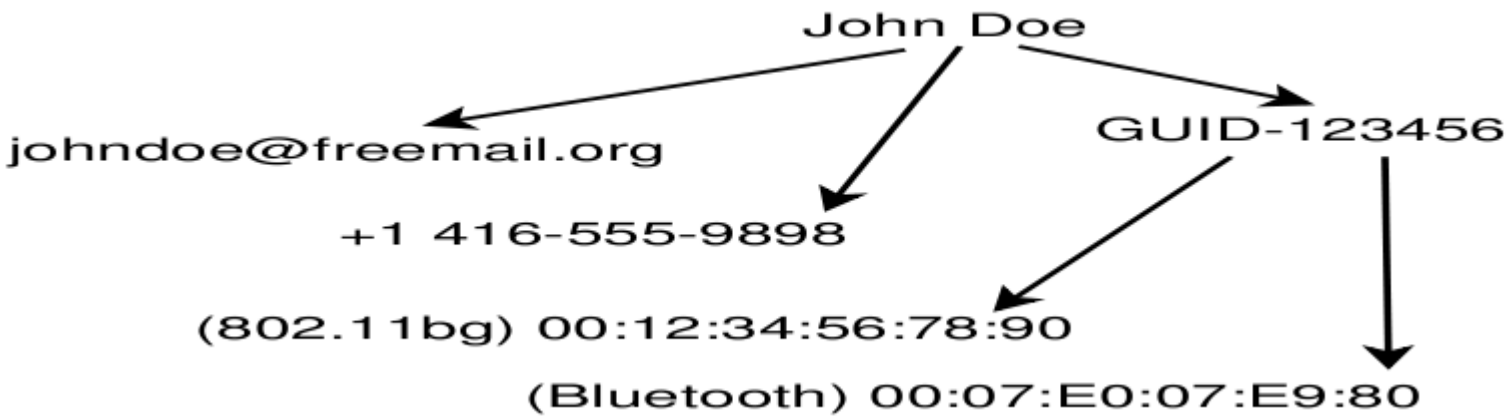
- Connectivity Manager
  - Initiates neighbour discovery, provides connections to neighbours, and estimates costs (money, energy, time) of transmitting objects
  - Connectivity is regarded as a schedulable resource, and must be delegated to the resource manager for scheduling
  - User-defined policies may be used to manage multiple interfaces
  - Prototype developed for 802.11 standard, but multiple interfaces may be supported

The Haggle Architecture (3)

- Protocol and Forwarding Manager
  - Encapsulates all methods of transferring data objects, e.g. SMTP, POP, and a direct P2P protocol; marks names as “nearby”
  - Appropriate protocol is selected just-in-time to perform the necessary steps to send and receive data
  - Forwarding algorithms determine the suitability of a next-hop for the transmission of application and user-level messages
  - Support for using, at the same time, several routing algorithms (Distance Vector, Epidemic, etc)
  - Implemented: Direct (send to a neighbor) and Epidemic

The Haggle Architecture (4)

- Names: Name Graphs



- Allows apps to specify and access trees of names, e.g. “John Doe” is a parent node for his laptop’s MAC address and for his email address
- One individual can have many different addressable identities, reachable using different connectivity methods
- Any name can be an address if there's a protocol capable of understanding it

The Haggle Architecture (5)

- Data Management and Data Objects
  - Haggle's data format is designed to be searchable and structured
  - Data and Metadata are clearly exposed to applications
  - Relationships between application data units (e.g, webpages and its images) should be representable
  - Data Objects(DOs): atributes consisting of a pair of type and value
  - Links between DOs is a directed graph: photoalbum+metadata, email+attachments
  - Object filters allow searching for objects using regular expressions: (mimetype=text/html)

The Haggle Architecture (5)

- Data Object example

Message

DO-Type	Data
Content-Type	message/rfc822
From	Bob
To	Alice
Subject	Check this photo out!
Body	[text]

Attachment

DO-Type	Data
Content-Type	image/jpeg
Keywords	Sunset, London
Creation time	05/06/06 2015 GMT
Data	[binary]

The Haggle Architecture (6)

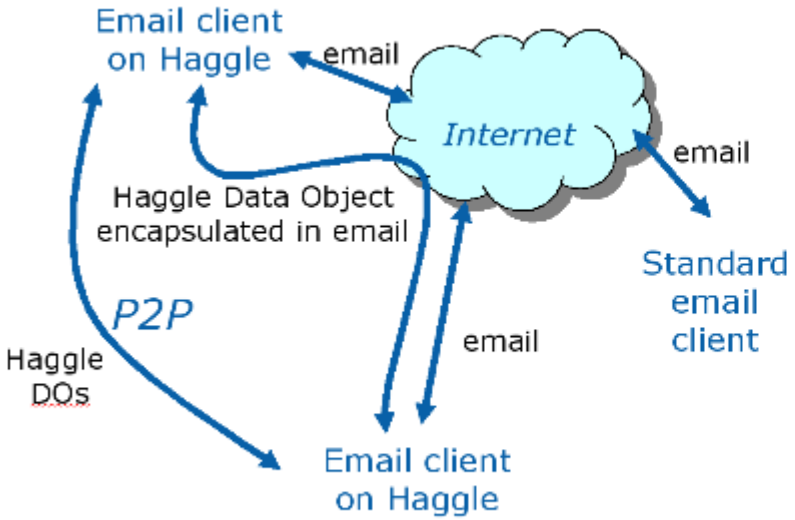
- Scheduling and Managing resources
  - Requests for network use from components and applications are delegated to the Resource Manager (RM)
  - RM considers which tasks are allowed to execute by evaluating whether it's beneficial and cost-effective according to user's preferences and policies
  - RM calculates how best to use each connectivity at each time point based on the benefits and costs of possible actions

Case studies

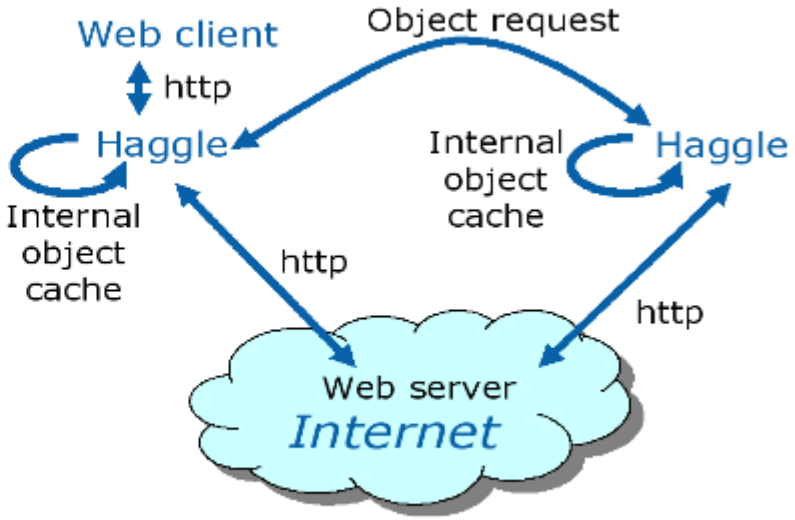
- Email on Haggle
  - Adaptation of Email applications (minimal reconfiguration) to work on Haggle, enabling email to be sent in a peer-to-peer fashion
  - Haggle implements localhost SMTP/POP proxy as Haggle native applications
  - Users need to setup a proxy in their email application to that of Haggle's proxy
  - Emails are received by the proxy and translated into Data Objects
  - The proxy then uses the recipient field to search for an appropriate **name** which describes the intended recipient

Case studies

- Email on Haggle



(a) Email Application



(b) Web Application

Case studies

- Email on Haggle
  - Then, a forwarding request in created to send the mail object to the individual described by the name object
  - Haggle can then decide
    - When the message will be delivered
    - The protocol to use
    - Which network interface to use
  - Decision to use infrastructure or peer-to-peer is up to the availability of infrastructure (i.e, Internet access) or **throughput improvement** (thus using a peer-to-peer connection)

Case studies

- Web surfing on Haggle
  - Haggle implements a Web proxy
  - If Internet is available, then it proxies connections normally
  - Caches are built locally by the Data Manager from the Internet, in each node
  - If Internet access is unavailable or too expensive to use, the proxy creates a filter subscribing to the URL of the requested web page
  - The request is sent over the Haggle network in the same fashion as the email access, but using the HTTP proxy

References

- http://www.haggleproject.org