



An Extensible Network Resource Abstraction for Applications on Mobile Hosts

Arjan Peddemors♦*, Ignas Niemegeers* and Henk Eertink♦

COMSWARE'07, January 11th, 2007

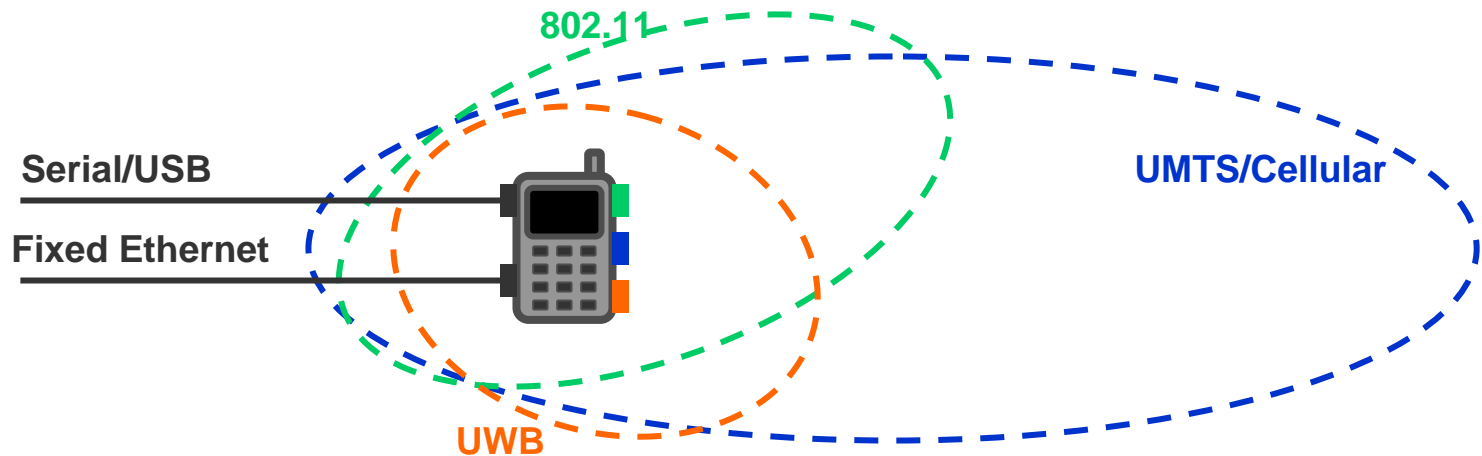
♦ INCA Group
Telematica Instituut
Enschede, The Netherlands

* WMC Group
Delft University of Technology
Delft, The Netherlands



Multiple and Heterogeneous Network Resources on Mobile Devices

- Reality today: mobile devices with multiple network interfaces capable of carrying IP packets



- Network resources on mobile devices
 - Offering multiple links and paths
 - Heterogeneous technologies
 - Dynamic state (inherent to mobility)

Usage of Network Resources on Mobile Devices

- Hiding link layer and network layer aspects is for many mobile applications not acceptable
- Actions of interest to applications
 - Detection and scanning
 - Link Activation
 - Path selection for data traffic
 - Monitoring
- Type of information and level of detail depends on the application
 - WWW Browser
 - Remote Control
 - Video Newsfeed Player
 - Media Stream Player



Objectives

To provide an *extensible abstraction* that supports applications on mobile devices with their decisions on the *usage and manipulation of network resources*

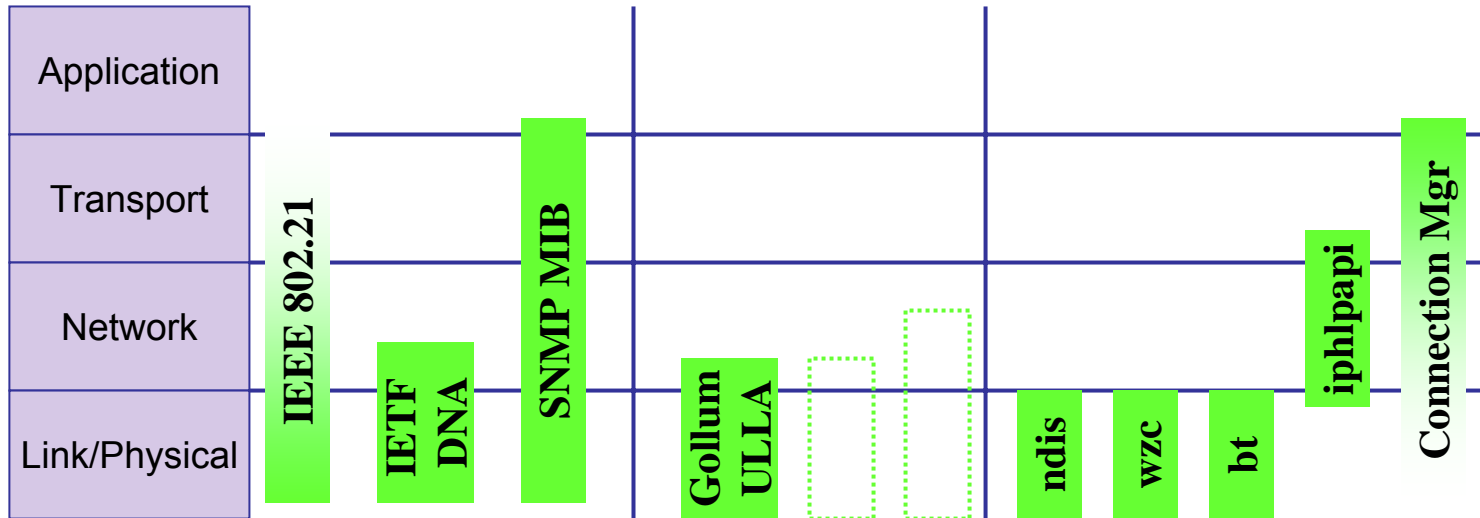
- Presenting cross-layer network resource information “as-is”
- Capable of reflecting what’s typically available on personal mobile devices
- Offering flexible levels of abstraction to match the decision making needs of the application
- Simple and lightweight

What's already there...

Formal cross-layer models and standardization

APIs dealing with network heterogeneity

Regular Operating System APIs (e.g. for Windows CE)



Contributions

- Network Resource Model (NRM)
 - Providing description of cross-layer network resource entities and their inter-relationships
 - Extensible through type hierarchies
 - Used by applications for network resource awareness and control
- Network Abstraction Layer (NAL)
 - System facility that provides dynamic network resource information – based on the NRM – to applications
 - Extensible through plugins to accommodate addition of new technologies (e.g. new wireless link)
- Experiments and Validation (running code)

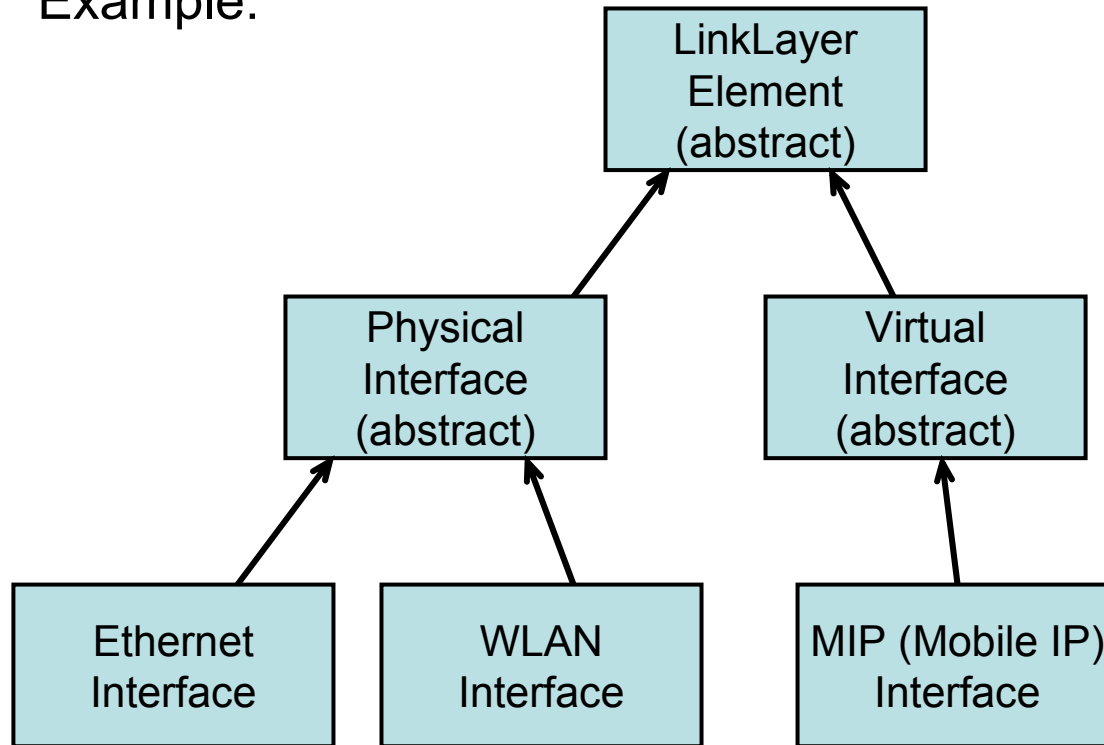


Topics

- Introduction, Motivation and Objectives
- Network Resource Model (NRM)
- Network Abstraction Layer (NAL)
- Experiments
- Conclusions and Future Work

Network Resource Model

- Network resource description with various levels of abstraction: enabled through type hierarchies
- Example:



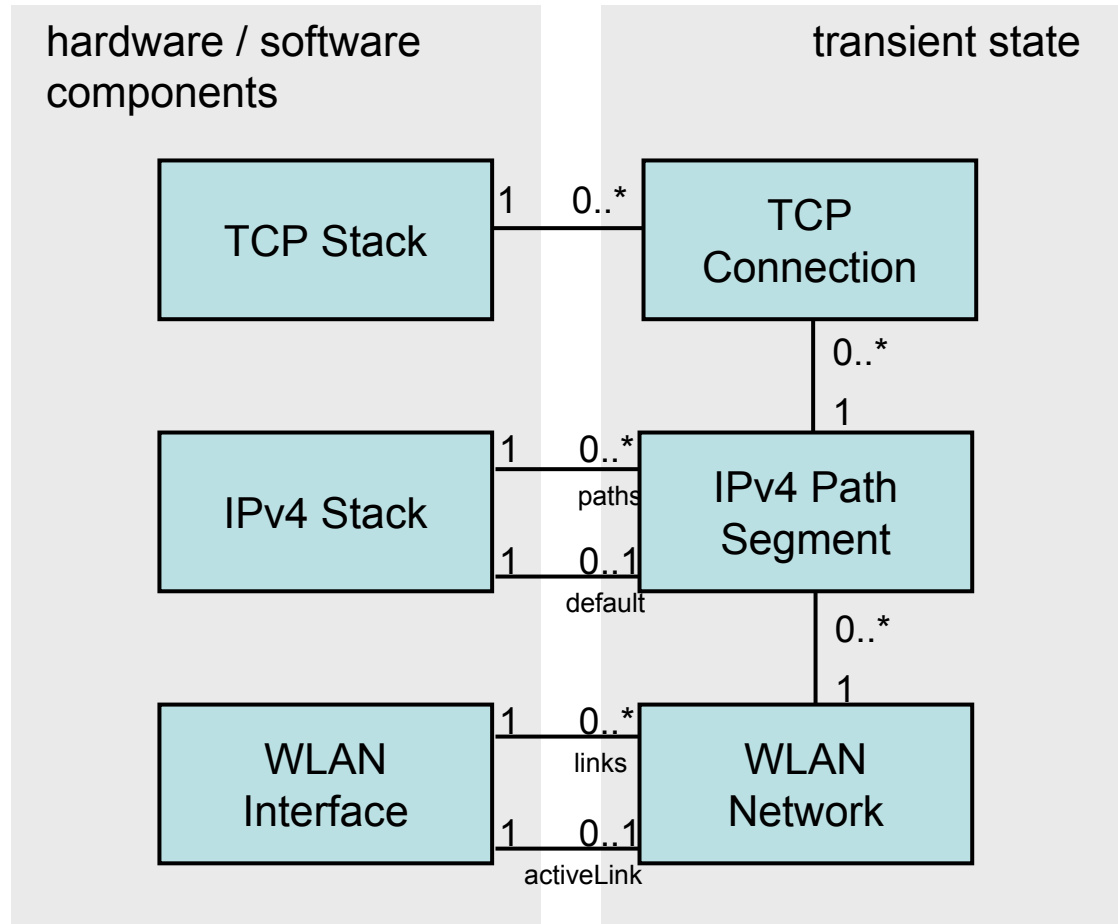
Network Resource Model

- Abstract types define common characteristics and common operations
- More specific types define characteristics and operations that apply to the specific type
- **Fully cross layer:** covering entities at Link, Network and Transport Layer
- Single inheritance type hierarchy
- Every object can be uniquely identified by a numerical identifier (defined per generic type)
 - E.g. linkId, pathSegmentId,

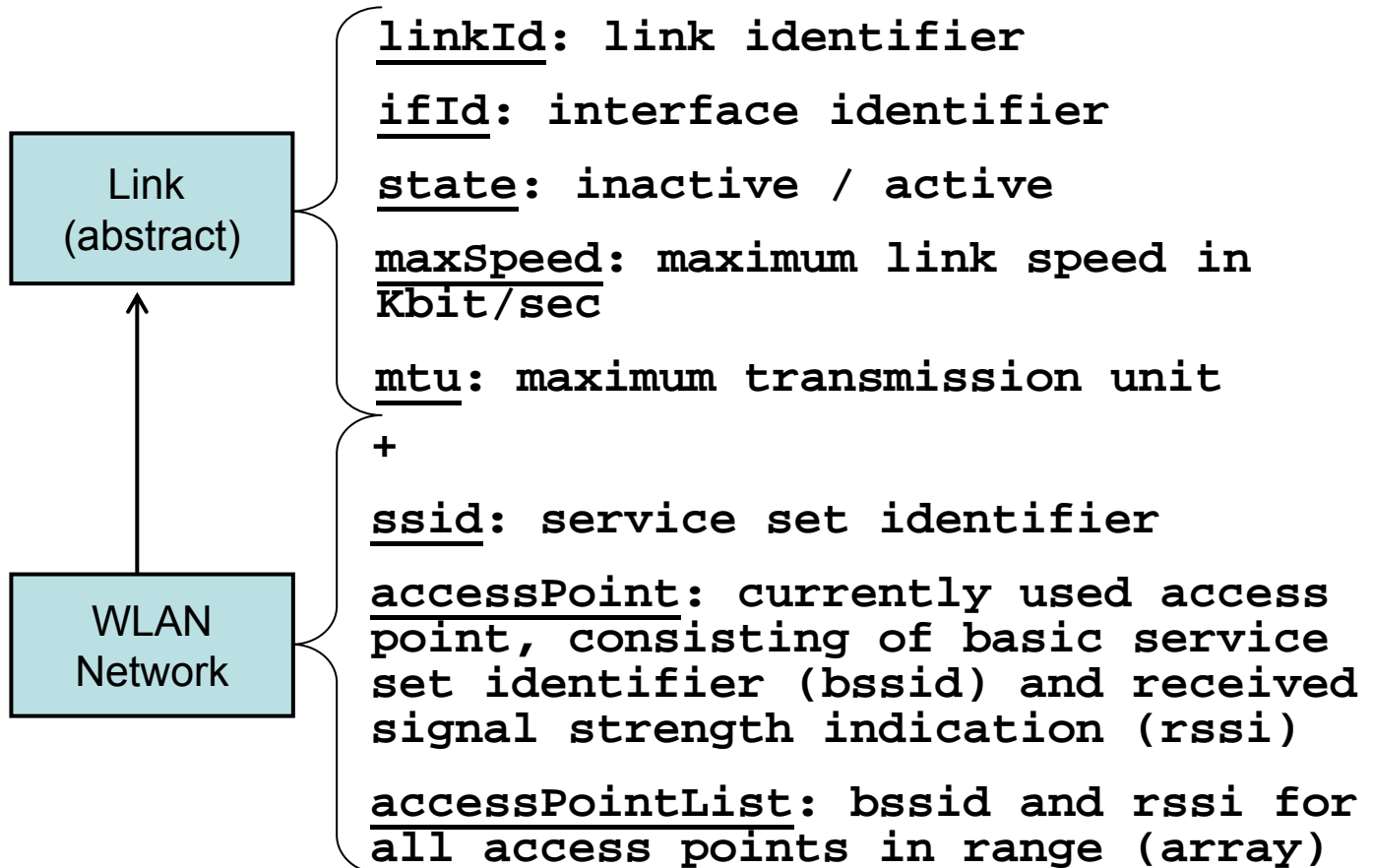


Network Resource Model

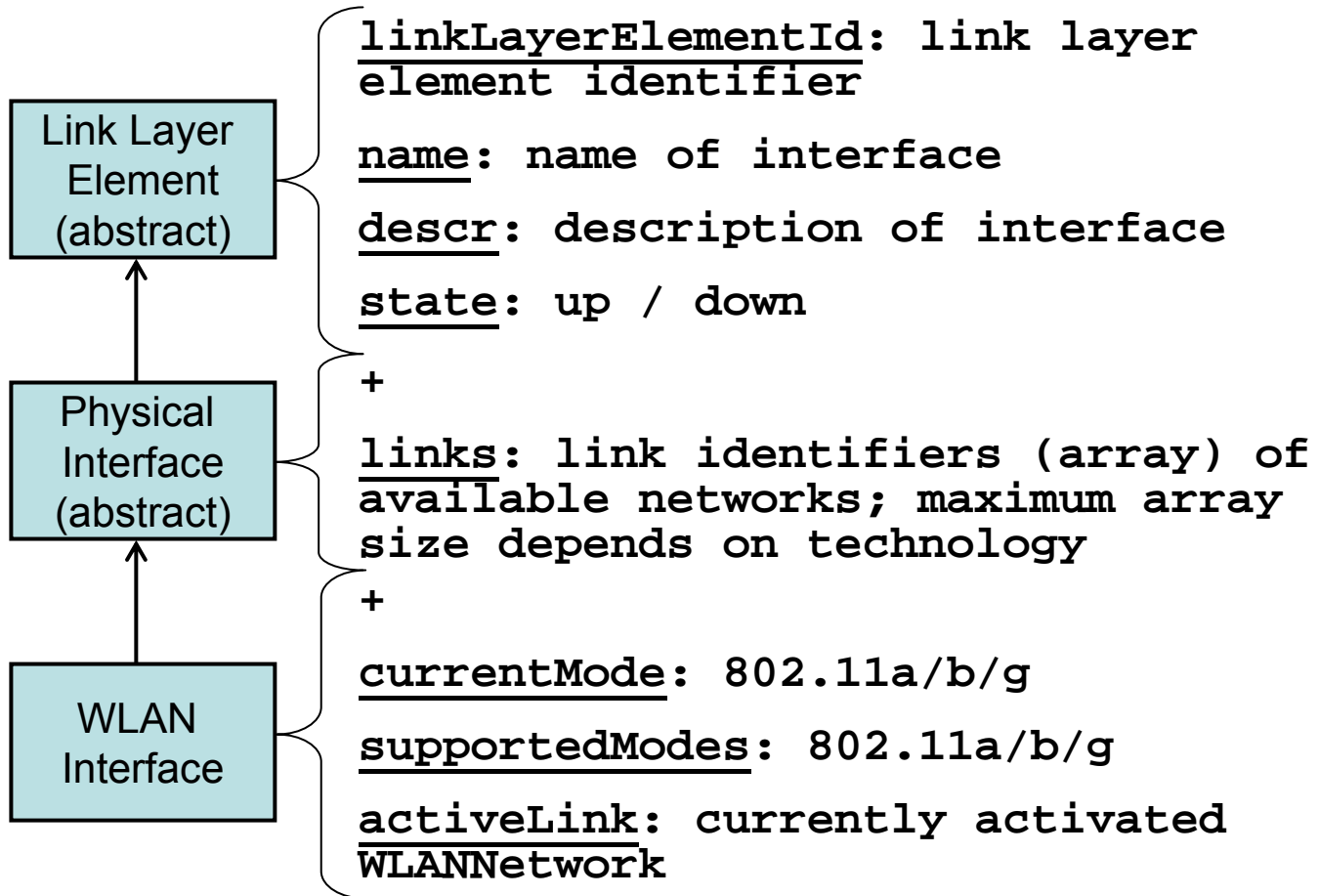
Inter-relationships of a few well-known types



Network Resource Model: Example Characteristics



Network Resource Model: Example Characteristics



Network Resource Model Schema / Instance Example

```
<xs:complexType name="BaseStation">
  <xs:sequence>
    <xs:element name="cid" type="xs:unsignedInt"/>
    <xs:element name="lac" type="xs:unsignedInt"/>
    <xs:element name="rssi" type="xs:int"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="CellularNetwork">
  <xs:complexContent>
    <xs:extension base="Link">
      <xs:sequence>
        <xs:element name="operatorName"
          type="xs:string"/>
        <xs:element name="operatorNumber"
          type="xs:unsignedInt"/>
        <xs:element name="baseStation"
          type="BaseStation"/>
        <xs:element name="baseStationList">
          <xs:complexType>
            <xs:sequence minOccurs="0"
              maxOccurs="unbounded">
              <xs:element name="baseStation"
                type="BaseStation"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

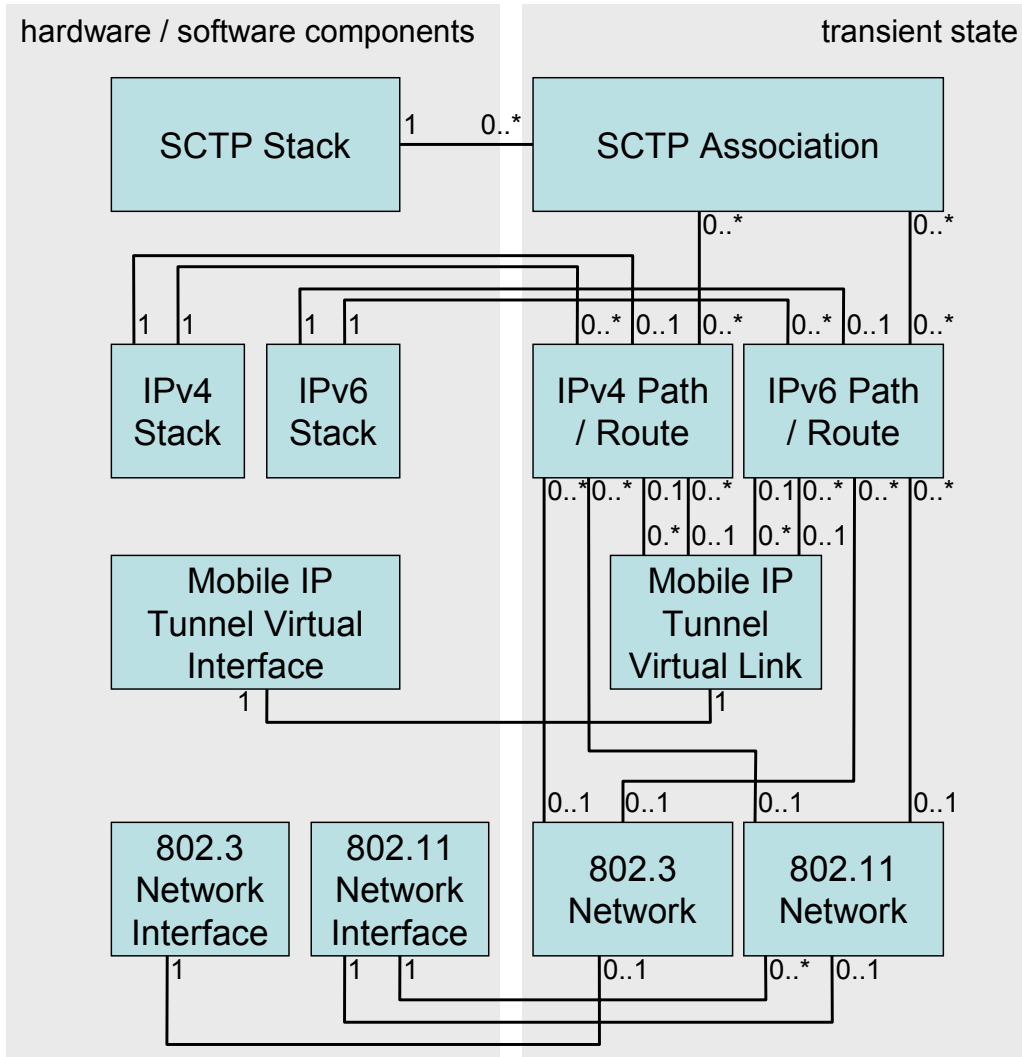
```
<WLANNetwork>
  <linkId>12</linkId>
  <ifId>2</ifId>
  <state>1</state>
  <maxSpeed>54000</maxSpeed>
  <mtu>1500</mtu>
  <ssid>GASTTI</ssid>
  <accessPoint>
    <bssid>00:0a:b8:1b:e7:70</bssid>
    <rssi>80</rssi>
  </accessPoint>
  <accessPointList>..</accessPointList>
  ..
</WLANNetwork>

<CellularNetwork>
  <linkId>15</linkId>
  <ifId>3</ifId>
  <state>1</state>
  <maxSpeed>28</maxSpeed>
  <mtu>1500</mtu>
  <operatorName>Vodafone NL</operatorName>
  <operatorNumber>20404</operatorNumber>
  <baseStation>
    <cid>7143</cid>
    <lac>42</lac>
    <rssi>-51</rssi>
  </baseStation>
  <baseStationList>..</baseStationList>
  ..
</CellularNetwork>
```



Network Resource Model

A more elaborate example

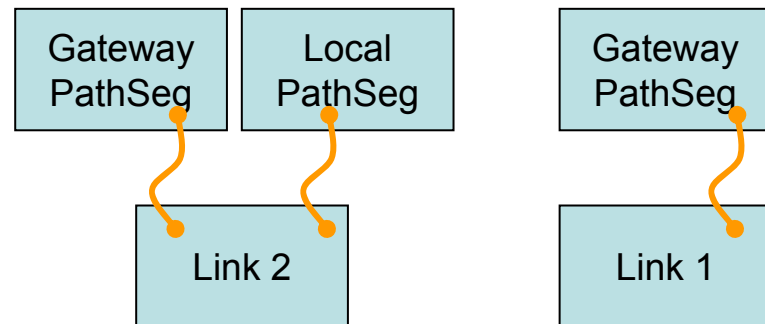


Example: Dynamics in Network Resources

Application View

- The application wants to reach other end points on the internet
 - Is primarily interested in available outgoing paths and their characteristics
- Only needs to know about *Gateway Path Segments* and *Links*
 - Gateway Path Segment indicates available outgoing path
 - Link provides characteristics such as maximum throughput
- This view is considerably more simple than the total view
 - In many cases, application may only need to monitor a few entities to have sufficient input for decision making

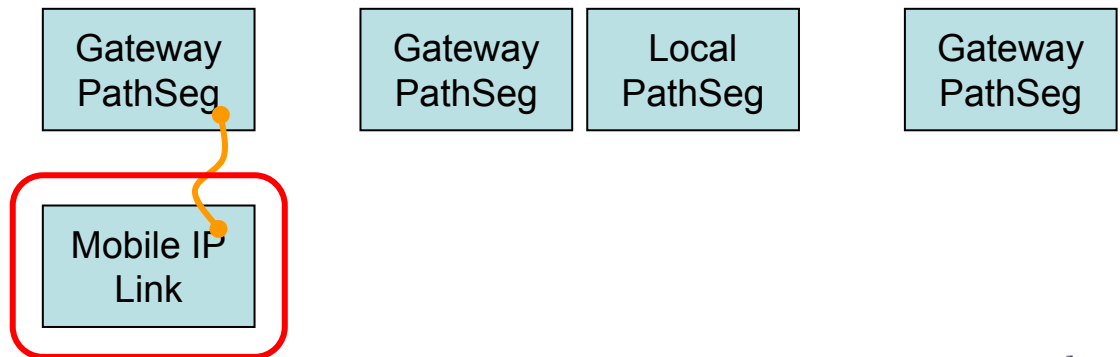
Now, the application can choose between two paths to setup a transport to another end point



Example: Dynamics in Network Resources Application View with Mobility Management

- Now, suppose the application wants to reach other end points on the internet using a mobility management facility such as Mobile IP
 - Is only interested in outgoing paths over the Mobile IP virtual link, and the characteristics of this link
- Only needs to know about *Gateway Path Segments* using the *Mobile IP Link* (which is a virtual link / tunnel)
 - Mobile IP Link reflects characteristics of underlying real links
- Again, this view is considerably more simple than the total view

Application monitors MIP Link characteristics to make decisions on generation of network traffic

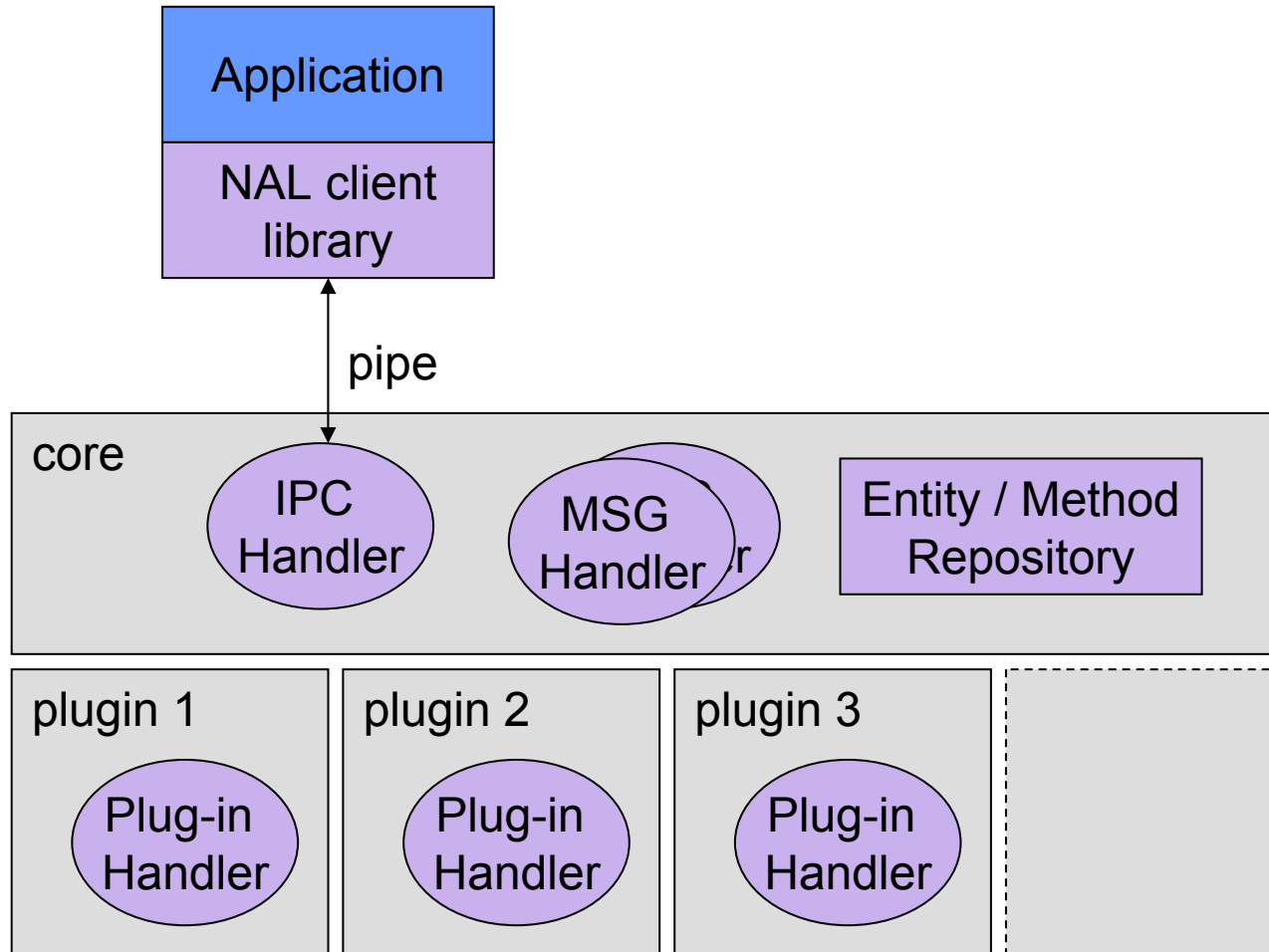


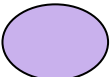


Topics

- Introduction, Motivation and Objectives
- Network Resource Model (NRM)
- Network Abstraction Layer (NAL)
- Experiments
- Conclusions and Future Work

Network Abstraction Layer Design



 = thread



Network Abstraction Layer Design

Core Functionality


- Inter Process Communication (IPC) Handler
 - Listening for incoming connection
 - Client connection is full duplex pipe that carries messages between client and NAL and vice versa
 - Dispatching client connections to Message Handlers
- Message Handler
 - Handles client request (subscriptions and commands)
- Repository
 - Holds the current Network Resource state
 - Keeps track of subscribers
 - Keeps track of registered commands



Network Abstraction Layer Design

Core Functionality

- Subscription
 - Various levels of granularity
 - **NAL_SUBSCRIBE_LAYER**: get info on all entities within one layer (link / network / transport)
 - **NAL_SUBSCRIBE_TYPE**: get info on all instances of a specific type
 - **NAL_SUBSCRIBE_INSTANCE**: get info on one specific instance
 - Different kinds of subscription may be combined (e.g. all from network layer + all instances of a link type)
- Command (IOCTL)
 - IOCTL style commands, to be executed on an instance of a type
 - Every type defines its own set of commands; generic commands for more abstract types, technology specific command for more specific types
 - Repository keeps track of defined commands



Network Abstraction Layer Design Extensible through Plugins

- All network technology specific functionality resides in plugins in the form of shared libraries
 - Easy customization to match the widely varying configurations of mobile devices
 - Allows for easy extension as new network technologies become available
 - Makes NAL more portable
- Plugin hook “**thread**”
 - Called when the plugin loads
 - Inside its own thread
- Plugin hook “**transaction**”
 - Called as part of every transaction executed on the repository
 - Can be used to keep repository state consistent



Network Abstraction Layer Design

Repository Operations

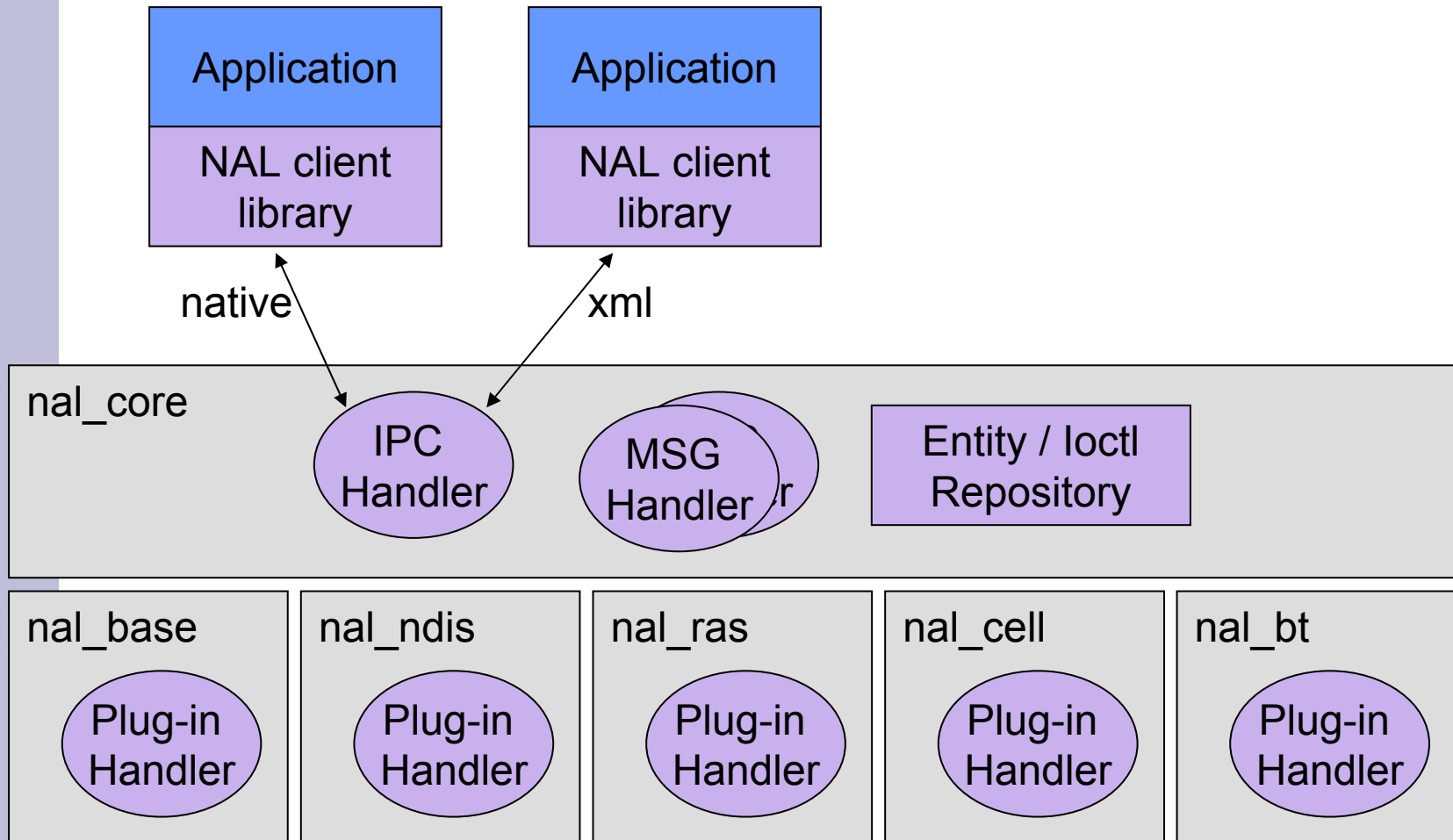
- All operations on the repository are executed in the context of a transaction
 - To keep the state as collected in the repository consistent
- Every plugin may initiate a transaction; all others are notified at the end through their “**transaction**” hook
 - Especially useful to establish relationships between layers
- Many common operations available to plugins
 - object create
 - object read
 - object update
 - object delete
 - iterators to read (part of) the repository

Topics

- Introduction, Motivation and Objectives
- Network Resource Model (NRM)
- Network Abstraction Layer (NAL)
- Experiments
- Conclusions and Future Work



Network Abstraction Layer WinCE Implementation





Network Abstraction Layer Implementation

C Language and Type Hierarchies

- NAL implementation is in C
- C is not an object oriented language although we use type hierarchies in the network resource model
- This conflict is resolved by using a simple mechanism based on “C struct nesting” combined with a type description for every available type
- For every type, the following actions are defined
 - init: initialize the instance
 - clear: clear the instance
 - pack: pack an instance for serialization
 - unpack: unpack a serialized instance
 - packlen: length in bytes of serialized form
 - xmlout: print instance as XML

Current NAL implementation is roughly 10000 lines of C code (.h and .c files) Implementation is tested on a Qtek 9090 (WM2003SE) and Qtek S200 (WM5)



Network Abstraction Layer Implementation

nal_core / nal_base

- nal_core
 - The full duplex pipe for IPC is implemented using Windows CE Message Queues
- nal_base
 - Covers all types on transport and network layer as well as a few 'standard' link layer types such as the localloop virtual interface
 - Triggers (mostly) on IP address changes and route table changes, using the Windows CE 'iphlpapi' API



Network Abstraction Layer Implementation nal_ndis

- Mostly based on the NDISUIO Windows CE API
- Currently defines the WLANInterface and WLANNetwork types
- If need arises, will define (fixed) EthernetInterface and EthernetNetwork in the future
- Gathers info on
 - All available 802.11 networks: ssid, currently active network, access point used for current network
 - All available 802.11 access points: bssid and rssi signal strength



Network Abstraction Layer Implementation nal_ras

- Mostly based on the RAS Windows CE API
- Currently defines the SerialInterface and SerialNetwork types
- Gathers info on
 - All available “direct” serial interfaces (not modems and VPNs)
 - Covers the USB serial interface



Network Abstraction Layer Implementation nal_cell

- Mostly based on RIL interaction
- Currently defines the CellularInterface and CellularNetwork types
- Gathers info on
 - All available GSM/GPRS networks (operator name, operator number)
 - Signal strength and cell id of currently used cell (so, no info on other cells)



Network Abstraction Layer Implementation nal_bt

- Mostly based on the Microsoft Bluetooth set of APIs (Microsoft Bluetooth stack)
- Defines BluetoothInterface, BluetoothLink and BluetoothNode
- Gathers info on
 - (Discoverable) Bluetooth devices in range
 - Bluetooth devices used for PAN access
 - Bluetooth device address and user-friendly name

Network Abstraction Layer Implementation

Client side / GUI

- Libraries
 - Native library (nallib)
 - XML stream library
- Language bindings for
 - C#
 - Java(based on XML stream lib)
- Command line client in C (nalclt) using native lib

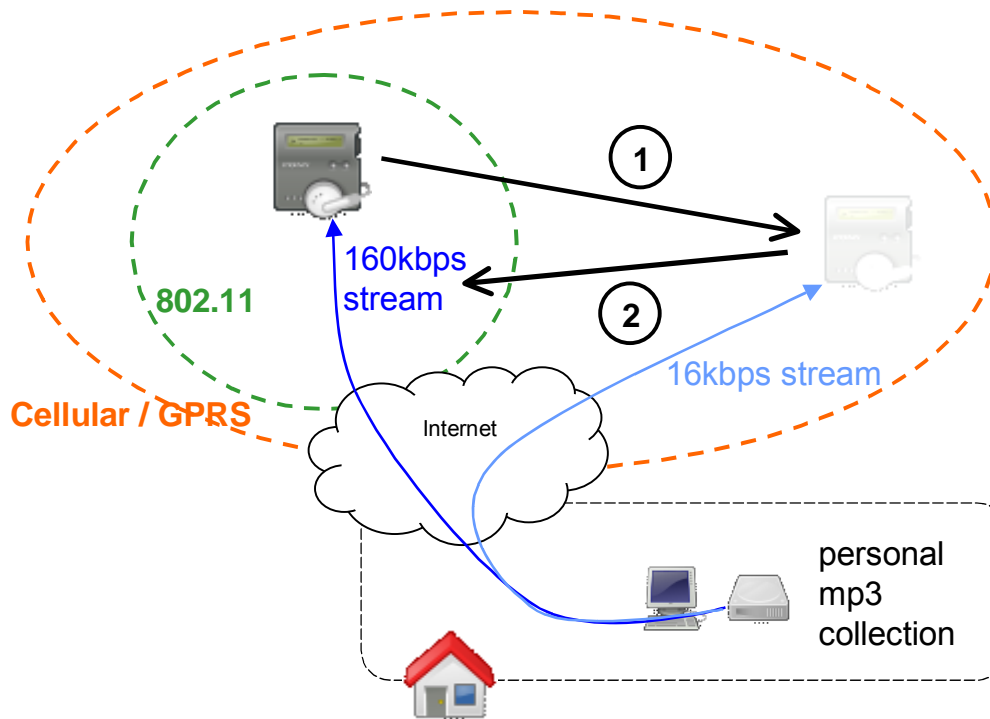


```
Command Prompt 17:24
type IPv4LocalPathSegment(131273), id
>>> delete
type IPv4GatewayPathSegment(131274), id
>>> transaction end
>>> transaction begin
>>> update
type: WLANNetwork(65838): Link(65540)
<WLANNetwork><linkId>19791873</linkId>
s</services><maxSpeed>11000</maxSpeed>
Point><bssid>00:0d:ed:90:1c:ff</bssid>
List><accessPoint><bssid>00:0d:ed:90:1
<accessPoint><bssid>00:0d:bc:68:32:23<
ssPoint><bssid>00:0d:bc:68:34:05</bssi
nt><bssid>00:0d:ed:90:1f:4a</bssid><rs
ssid>00:0d:ed:c3:0c:69</bssid><rssi>-7
WLANNetwork>
>>> transaction end
```

Experiments

“Networked iPod”

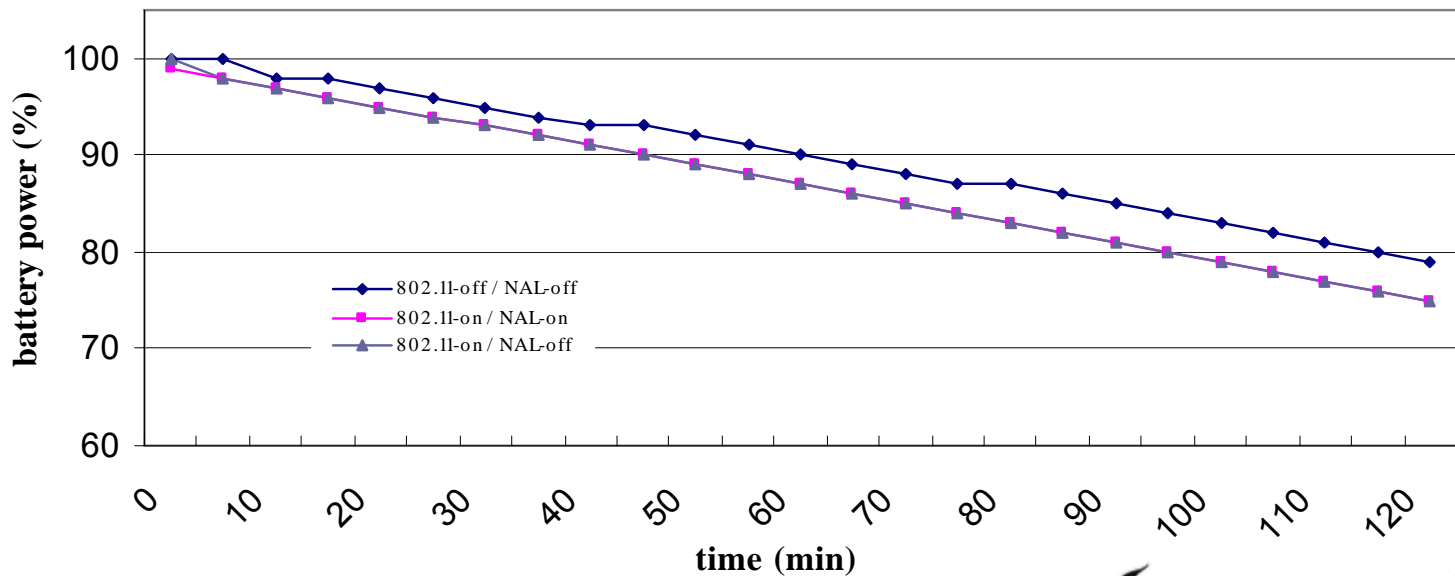
- Using abstraction offered by NRM / NAL to make decisions on handover from GPRS <-> 802.11



Experiments

NAL power consumption

- Power consumption of NAL as running on a Qtek 9090 device
 - Device is stationary
 - Measuring battery power percentage
- Very low overhead





Topics

- Introduction, Motivation and Objectives
- Network Resource Model (NRM)
- Network Abstraction Layer (NAL)
- Experiments
- Conclusions and Future Work

Conclusions and Future Work

- We have introduced
 - Network Resource Model for flexible awareness and control over network resources by mobile applications
 - Network Abstraction Layer system facility
- Experiments show
 - NRM / NAL can be implemented on unmodified existing mobile devices
 - NAL has a low power consumption
 - Abstraction is useful for streaming audio application
- Future work
 - Applying NAL to collect user mobility traces to optimize resource usage of delay tolerant applications
 - “*Experience-Based Network Resource Usage*”





Thanks!

Contact: arjan.peddemors@telin.nl

Website: <http://cosphere.telin.nl/nal/>
(NRM schema + NAL source code)

