Network Resource Awareness and Control in Mobile Applications

As mobile devices become increasingly capable of simultaneously using various kinds of wireless and fixed networks, developers must match mobile application needs to network resource dynamics. The authors' network resource model and network abstraction layer (NAL) software offer an extensible network resource abstraction to applications running on mobile devices. The NAL service can be used in conjunction with well-known APIs, such as the Socket API, which provide access to these resources; it also accommodates support for mobility-management facilities, such as Mobile IP, if available.

Arjan Peddemors, Henk Eertink, and Mortaza Bargh
Telematica Instituut

Ignas Niemegeers
TU Delft

Mobile devices have developed rapidly in the past several years as open computing platforms that can install and run software from different origins. Growing hardware capabilities, together with general-purpose operating systems and abundant possibilities for network communication, have made the mobile device platform an environment that enables many exciting applications.

The properties of network resources on mobile devices are dynamic in many situations. As users are mobile, they walk in and out of range of wireless networks, often experience fluctuations in throughput capacity, and obtain different levels of support for IP connectivity. Depending on their specific needs, mobile applications benefit from taking into account aspects of these dynamics — for instance, by adapting data rates to maximum available capacity or controlling when to scan and activate specific networks. Contrary to some approaches to mobility and roaming, we argue that network and link-layer aspects cannot be fully hidden from applications but must be exposed in a flexible manner.

To support applications in dealing with this cross-layer resource awareness and control, we have developed an extensible network resource model (NRM) that describes the available network entities and their interrelationships below the application layer. Applications can take different views on available network resources using the NRM such that they provide just enough information to match

1089-7801/07/$25.00 © 2007 IEEE

with the applications' decision making needs. They connect to the network abstraction layer (NAL) service on the mobile device to be notified of changes and to control available resources.

Mobile Devices and the Protocol Stack

In addition to the personal links they provide between users and devices, their continuous physical closeness and availability, and their battery-powered operation, mobile devices can exploit the availability of multiple and heterogeneous network resources. The primary difference between applications involved in data communication on mobile devices and those on more traditional computers is the difference between a dynamic and a relatively static network configuration — the protocol stack's layering has been a good match with the more static configuration. The elementary abstraction that more traditional hosts offer comes from the transport layer — which sets up and maintains connections with other endpoints — combined with a general notion of a binary IP connectivity state (that is, there's either connectivity or there isn't). This layering also fits well with the network hardware most often found on traditional computers: a fixed Ethernet interface with a (generally speaking) binary connectivity state. Applications set up end-to-end connections with another host and aren't concerned with the settings and configurations of entities at lower layers (network, link, or physical), which establish and maintain this connection.

Although this layered approach has served us well over the years, the situation is different in a mobile setting: wireless network hardware can be in any one of multiple modes, such as "off," "sleep," "power safe," or "fully on," and can detect multiple networks with different and fluctuating capacity, all of which is inherent to mobility. Network hardware might be able to sustain connectivity to more than one network, and mobile devices are often equipped with multiple network interfaces. Today, a mobile device configuration with three or four interfaces (for example, Bluetooth, 802.11, cellular, and fixed serial such as USB) isn't exceptional. This means applications on mobile devices can choose available or potentially available data networks to carry traffic. Many factors influence this decision, such as capacity, cost, security, power usage, and so on. Moreover, different applications (email, Web browsing, file download, streaming media, and peer-to-peer clients, for example) might have different needs, perhaps even varying in time, each requiring their own level of control over network resources. They might use various kinds of mobility-management mechanisms — such as Mobile IP at the network layer or Mobile Stream Control Transmission Protocol (SCTP) at the transport layer — to at least partially hide the attachment and detachment of consecutive networks while the user roams.

With this in mind, the simple transport layer abstraction we've been using in traditional computing equipment no longer suffices for all cases. We must make mobile applications aware of the state of entities at the network layer and below to let them control, for example, whether connection traffic goes over wireless network A or B. However, we also need to make sure that this information has the right level of detail: just enough to make decisions, providing a high-level view that doesn't reveal to the application all information at all layers.

Network Resource Abstraction

Let's consider several examples that illustrate how much awareness and control various mobile applications require. A Web browser uses HTTP over short-lived TCP connections to retrieve Web pages from a server, but it isn't necessary to hand over these connections while roaming because users can easily resolve an accidental disturbance with a page reload. So, the browser doesn't really care which underlying network it uses; for efficiency, however, it might want to establish TCP connections without mobility management.

The situation is different for applications that provide control over appliances in the user's surroundings, such as audiovisual equipment, because the appliances are reachable through their own or separate in-building wireless network facilities. The remote control application initiates control connections as soon as the mobile device is within those networks' reach. This means that it must tell the operating system to occasionally scan for available networks and activate those that allow connections to known appliances. In some situations, this might be the system's default behavior, but when network scanning and link activation is expensive in terms of power consumption — the case for current 802.11 technologies — networks will be activated only on request. This makes the application more directly involved in below-IP-layer operations.

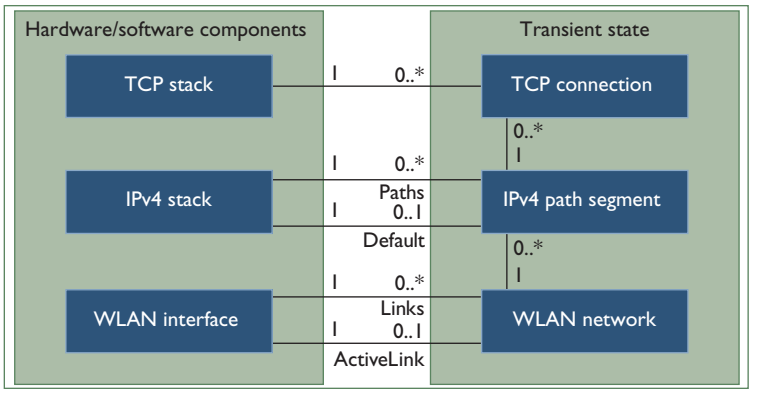


Figure 1. Several well-known network resource model types and their interrelationships. The relationships' cardinality captures basic technology characteristics. For example, the WLAN interface has two relationships with possibly multiple WLAN networks: one to indicate the WLAN networks in range (zero or more) and one to indicate the currently active network (zero or one). Other wireless technologies might allow for multiple active connections.

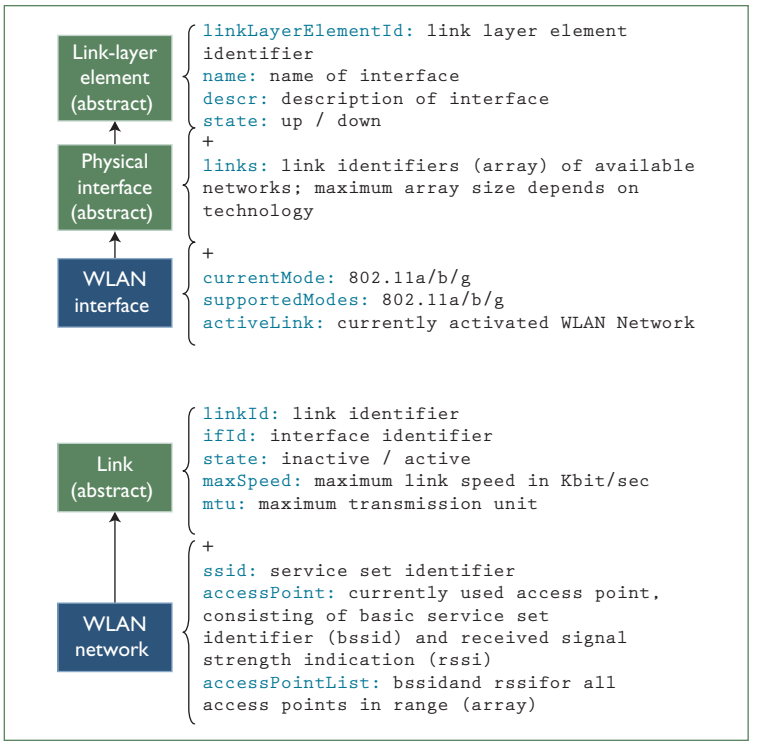


Figure 2. Two examples of a type hierarchy in the network resource model. Abstract types define more generic characteristics, whereas concrete types define technology-specific characteristics. Applications can choose the level of abstraction by treating entity descriptions as more generic or more specific by selecting a type in the hierarchy.

A third example is a streaming media player that adapts the amount of data that passes over the connection with the server based on the available throughput while the user roams. The different networks that come and go over time each have their own bandwidth characteristics: it's hard to know the actual available throughput, but the limit imposed by currently available network technology provides useful hints — there might be three orders of magnitude between cellular and WLAN access, for example. The player doesn't need to know technology-specific characteristics, but uses the speed characteristic defined for all technologies. Even if Mobile IP manages the stream connection, knowing the underlying network's throughput characteristics is still valuable. In general, applications need varying degrees of control and information — from almost nothing to highly technology-specific — covering network detection and scanning, network activation, and data-traffic generation.

When we started trying to provide better support for network resource adaptation in mobile applications, we realized that two basic aspects were lacking. First, we needed a mechanism that let applications look further down the protocol stack through a view that matched their needs. Second, we needed a formal model to describe the network resources in a cross-layer fashion that could produce these flexible views. Such a model needed to capture essential characteristics of entities at several layers and, equally important, indicate the relations between them. Surprisingly, we found that such a model didn't exist: the *management information base* (MIB) types[1] do define cross-layer entity relations, but the scope and objectives are quite different; for instance, they don't support a wide variety of wireless networks. Thus, we defined our own NRM. (For more discussion of other work in this field, see the "Related Works in Network Resource Usage on Mobile Devices" sidebar).

The NRM considers entities at three different layers of the protocol stack: the link layer, including everything below; the network layer; and the transport layer. At the (physical) link layer, bits and frames are transferred over links, which a network interface maintains. An example is the link setup between a host's Ethernet interface and an Ethernet switch. At the network layer, network nodes send packets over paths that consist of a concatenation of links. At the transport layer, data travels over end-to-end transports, which use one or multiple paths.[2]

Each of these fundamental entities is managed by the hardware and software functionality at the respective layers; network interfaces maintain links, IP stacks maintain paths, and transport-layer stacks

maintain transports (for User Datagram Protocol [UDP], Transmission Control Protocol [TCP], Stream Control Transmission Protocol [SCTP], and the Datagram Congestion Control Protocol [DCCP]). At the transport layer, and especially the network layer, the available entity types are relatively few, but at the link layer they can be numerous, depending on how many network technologies need to be incorporated. The types are defined following an object-oriented paradigm resulting in type hierarchies: basic and abstract types define generic characteristics, whereas deriving types define more specific characteristics. Note that the NRM doesn't exclude ad hoc networking; such networks can simply be modeled with links and paths. Although here we define only those link-layer technologies that are present in currently available devices, we can extend the NRM to form a broad taxonomy, through inheritance and by introducing new top-level types. Figure 1 presents some example types and their interrelationships, and Figure 2 shows two examples of type hierarchies.

Our abstract model covers only those segments of paths (single hop) that are directly connected to the host, not full end-to-end paths, because a multihop path isn't completely visible from the host's standpoint. Thus, the host can't uniquely identify an entire multihop path; the host can, however, provide a good description of a path's first segment. For mobile hosts, the first hop in the path is often highly relevant, as it's mostly over a wireless link, which is often the path segment with the most restricted resources and also the one with the highest influence on usage cost, at least from the user's perspective. So, information about the first hop of available outgoing paths might, in many situations, provide valuable information to applications generating network traffic. Additionally, the path's first segment is the only part that mobile hosts select. Typically, NRM path segments coincide with the entries found in the operating system's routing table. In the NRM we define gateway path segments for multihop paths to other Internet nodes and local path segments for single-hop paths to hosts on local networks.

The NRM is notated in XML schema (www.w3.org/XML/Schema); network resource descriptions that adhere to the model can be expressed as XML documents with structures dictated by the model's XML schema. We use various XML schema mechanisms to describe the model: abstract types, deriving (concrete) types, keys and key references to enforce uniqueness of identifiers, and so on. Figure 3 shows a small part of our model's integral schema; the rest is available from http://cosphere.telin.nl/nal/. Alternatively, applications can get network resource descriptions by means other than via XML documents, given that XML isn't the preferred format for all types of applications. We merely use the NRM schema as a formal notation to define the model: it can have different "bindings" in the form of XML, C, scripting, Java, and so on. Figure 4 shows a snapshot of a mobile

```
<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>
```

Figure 3. Network resource model types for describing cellular networks in XML schema. The CellularNetwork type derives from the abstract type `Link` *(not included here) and indicates the currently used base station (*`baseStation`*) as well as the list of all this operator's in-range base stations (*`baseStationList`*). The elements* `cid`*,* `lac`*, and* `rssi` *in* `BaseStation` *represent, respectively, the cell ID, location area code, and received signal strength indication of a base station.*

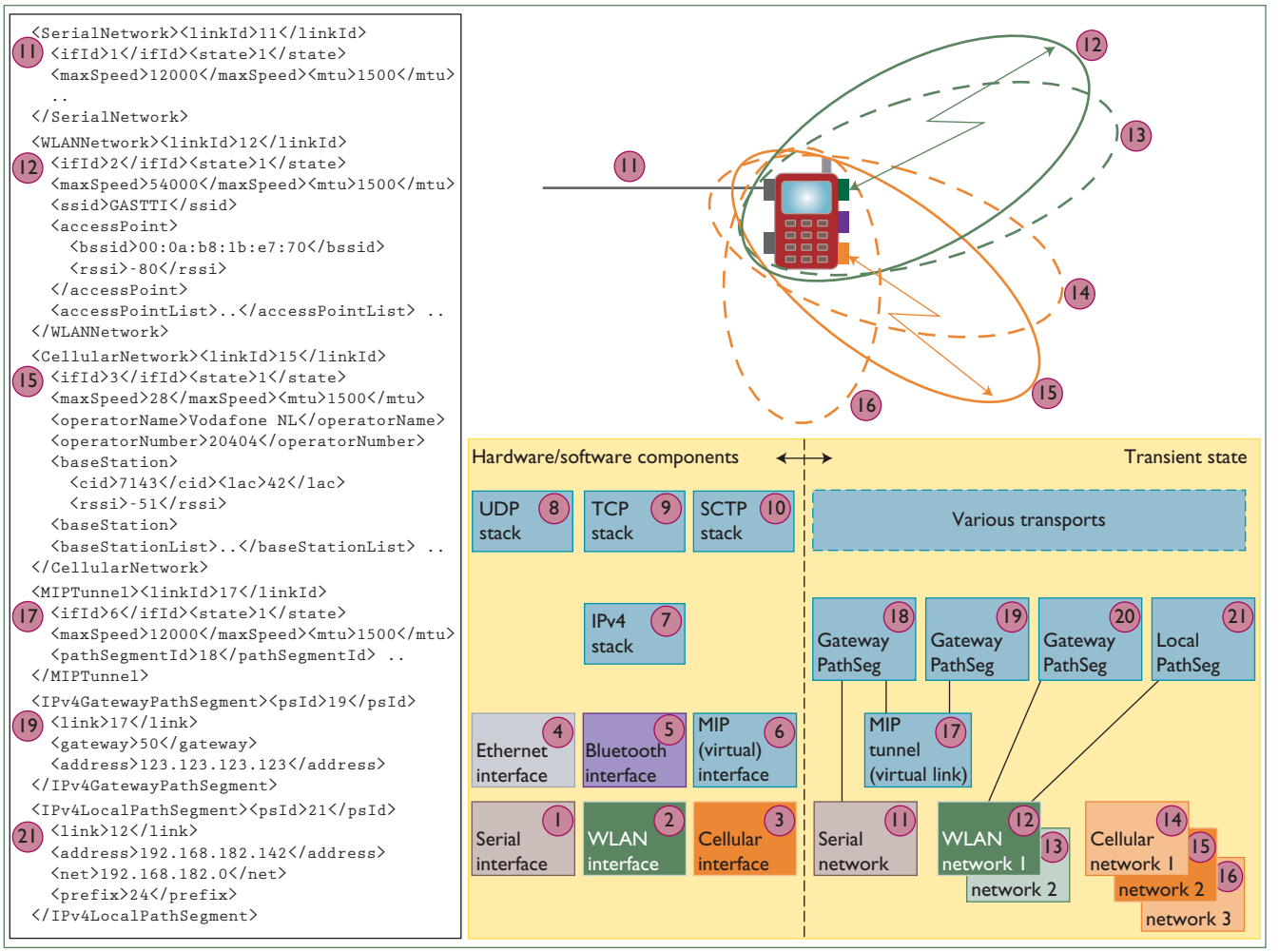


Figure 4. An example network resource description. This description uses the network resource model for a mobile device equipped with five physical network interfaces (fixed serial, WLAN, cellular, fixed Ethernet, and Bluetooth), as well as Mobile IP for mobility management. The Bluetooth and Ethernet interfaces don't have active links, but the serial link is up, the WLAN interface sees two networks (one actively connected), and the cellular interface sees three networks (also with one connected). Each entity description has an identifier, indicated as circled numbers, which the application uses to link the entities. The left side shows a partial description in XML. The right side shows the mobile device with five network interfaces and in-range networks (top) and the network resource description as a diagram (bottom). Note that the Mobile IP tunnel both uses and provides an IPv4 gateway path segment, allowing the tunnel packets to go over the serial link.

device with five network interfaces in range and attached to multiple networks.

Application Interaction

In addition to types that describe entities at the various layers — applied for actual resource description — the NRM also defines types that we use to define the interaction between applications and the (system) service that provides the resource description. This interaction consists of three kinds of messages:

- subscribe messages, which determine what resource information the system service sends to the application;
- I/O control (`ioctl`) messages, which let the application control network resources; and
- messages that describe the network resources themselves.

Application developers use subscribe messages in combination with type casts to construct a customized view of the available network resources' current state. Applications can make subscriptions on all entities in a layer, all entities of a type, or a specific entity of a certain type, as well as any of these in combination. So, a subscription filters out

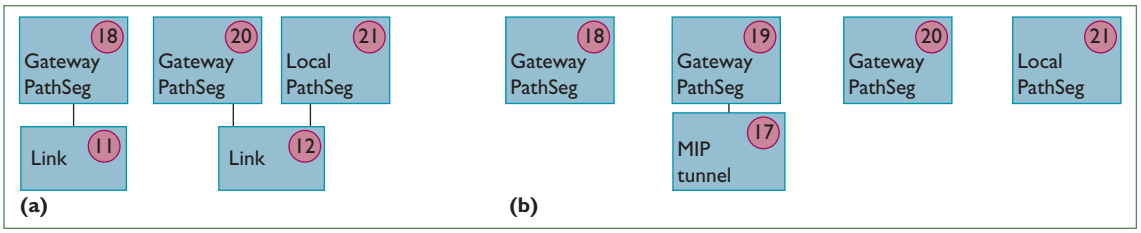


Figure 5. Two snaphots of network resources. They consist of only application-relevant entities for (a) determining possible paths and associating maximum throughputs, and (b) knowing the path supported by Mobile IP and maximum throughput for the Mobile IP tunnel.

those entities that are of interest to the application. Then, when applications receive resource changes, they can use the incoming entity description as-is or cast it to any of its super types to get a more general resource description. The `ioctl` message enables the application to manipulate the available entities — for instance, activate an available link through its interface or select the default gateway path segment (the default route).

With a resource change, the application receives messages that indicate that a new entity has been created or that an existing entity has been deleted or updated. These messages exist inside a transaction context, indicated with `transaction begin` and `transaction end` messages, such that at the end of a transaction, a consistent overall network resource description remains.

Now, let's look at what happens in a typical roaming scenario. Assume we have a mobile device with a configuration such as the one in Figure 4, with Mobile IP disabled, moving in and out of various networks' coverage and running an application that takes into account the maximum available throughput. With no mobility-management facilities available for TCP and UDP traffic, the application manages its own connections. It subscribes to link entities and path-segment entities and then obtains, at a certain moment, a resource snapshot such as the one in Figure 5a (ignoring the technology-specific link characteristics). The gateway path segments indicate a means of reaching other nodes on the Internet, and the links give the maximum throughput for these potential paths.

At first glance, managing its own connections in case of handovers might seem like a large burden for the application, but compared to the overall application-adaptation effort, it might actually be a limited one. Most of the work for a streaming player that adapts the stream content given a certain available maximum throughput might go toward codec changes, parameter renegotiation with the server, and so on, rather than into reestablishing connections.

Now, consider the same application with Mobile IP enabled. In this case, Mobile IP will maintain the connections as the user roams, so the application must know the path that the Mobile IP tunnel manages: it thus subscribes to the path segment and Mobile IP tunnel type (see Figure 5b), but still takes into account the maximum throughput, which changes as the tunnel moves from one physical network to another, reflected by an update of the tunnel entity (number 17 in the figure).

Different kinds of applications can use the NRM abstraction in different ways. The examples in Figure 5 show that even in a more elaborate network configuration, a designer can create a highly simplified view that captures information relevant to the application's decision process regarding network resource usage.

We'll note here that the application designer could use all this information in conjunction with existing network APIs, such as the Socket API. Most network stack implementations can be configured such that binding a socket to a local IP address — via the Socket API — determines the link over which the outgoing traffic is directed. This means that *local address binding* lets an application choose, for example, to send packets for a TCP connection out over a link available through a physical interface with certain desired maximum throughput characteristics, or over a Mobile IP tunnel and interface that sustains the TCP connection even through sporadic network availability. An application, however, can make such a decision only when it has received information that associates a local IP address with a link and characteristics such as maximum link throughput — essentially, cross-layer information. Note also that the Socket API offers substantial possibilities for enforcing network resource usage, but it can't help obtain the cross-layer information that the designer uses to decide which network resources to use.

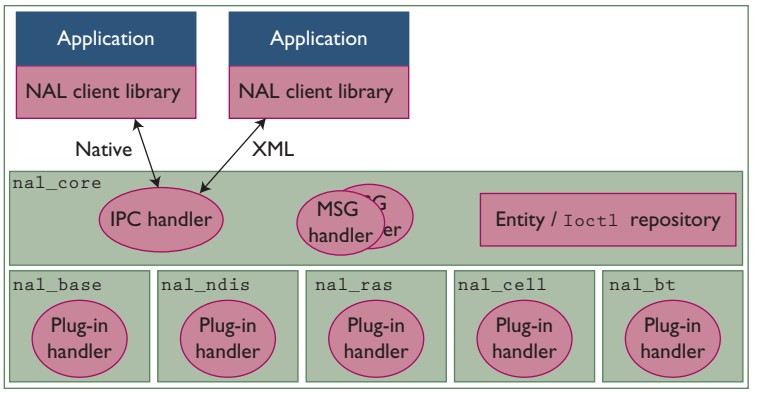
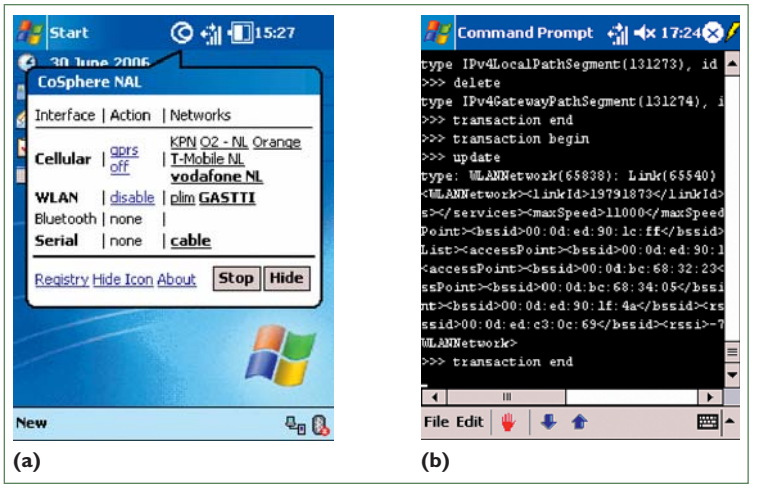


Figure 6. The network abstraction layer (NAL) system service. The service consists of several modules; the core one supports interprocess communication, message handling, and a network resource state and `ioctl` *repository. The other modules update technology-specific state information and handle* `ioctl` *calls. Applications can use native (marshalled C structs) or XML- (text-) based interaction with this service. Each ellipse is a separate thread.*



(a) (b)

Figure 7. The network abstraction layer (NAL) system. (a) The service GUI and (b) the NAL client. The NAL bubble shows a limited part of the full network resource description. The client shows deletes and an update as they occur inside transactions.

This observation, especially, led us to create our network resource abstraction.

Reference Implementation

We've employed a reference implementation of a service that provides network resource descriptions using the NRM. This system service might run on operating systems such as Symbian, Windows CE, or Linux, which are available on common mobile devices with multiple network interfaces. All these operating systems provide the means for obtaining information on network resources, with varying degrees of detail, and allow the application to manipulate these resources, mostly through a set of different APIs. Additionally, these OSs are general purpose and thus suitable for supporting many different applications.

The system service focuses on implementing a *network abstraction layer* (NAL) API that applications or other system services can use. Obviously, OS vendors might eventually implement such an API as a core OS facility; our objective here, however, is to experiment with our network abstraction, which is most conveniently implemented as an add-on component to an existing environment.

The NAL provides network resource information in a lightweight form and adheres to simple application interactions often found in system APIs. It implements the NRM-defined message interface with a native API in C and an interaction based on XML. The C API is best used for native applications in C and C++, whereas we can use the XML interface to provide bindings for languages such as Java. Furthermore, a binding might transform XML messages into language native objects using bridging technology.

Our NAL implementation (http://cosphere.telin.nl/nal) runs on the Windows CE operating system. It has plug-ins for abstract types and types for the network and transport layer (`nal_base`), 802.11 network types (`nal_ndis`), cellular network types (`nal_cell`), and USB fixed serial types (`nal_ras`) (see Figure 6). Additionally, we created an NAL command-line client that reflects all notifications the NAL generates (see Figure 7). The implementation consists of almost 11,000 lines of C code (`.c` and `.h` files) and has an executable size (`.exe` and `.dll-s`) of 90 Kbytes. The 802.11 plug-in gathers information on available networks, access points, and received signal strength indications (`rssi`). Per default, it obtains a list of access points every five seconds. Similarly, the cellular network plug-in retrieves information on in-range GSM General Packet Radio Service (GPRS) networks and operators and refreshes the currently used base station's `cellid` and `rssi` every five seconds. It also renews the operator list every five minutes. We provide a more detailed description of our implementation elsewhere.[3]

Experiments and Discussion

We ran the NAL in combination with the command-line client on a Qtek 9090 Windows CE device in a constant environment, in which the device didn't move, the cellular and 802.11 interfaces were acti-

vated, and no further applications ran ("NAL-on"). It consumed little extra energy compared to the same configuration that didn't run the NAL ("802.11-on"). As Figure 8 shows, both lines coincide; running the device without NAL and with the 802.11 interface switched off, however, shows decreased power consumption ("802.11-off"). In this two-hour period, the NAL produced around 40 notifications at startup (to provide a full snapshot of the available network resources) and slightly more than one update per minute for the rest of the run.

To validate the abstraction offered by the NRM as well as to test the NAL reference implementation, we executed several experiments with an implementation of a streaming audio player. On the server side, we based the test configuration on a modified streaming audio application called SlimServer (www.slimdevices.com), which lets users stream audio content from their personal collections to a node on the Internet. The streaming client itself is a modified GSPlayer (http://hp.vector.co.jp/authors/VA032810/) for the Windows CE operating system. Neither the SlimServer nor the GSPlayer software can maintain streams in the event of network handovers.

The test environment comprised a server machine storing a music collection and running the SlimServer software and a GSPlayer on a Qtek 9090 Windows CE device with, among others, a cellular GPRS interface and an 802.11b network interface. The device could reach the server either through the cellular IP network or through a local 802.11 network with a single access point (see Figure 9). No mobility-management protocols were available: in the event of network resource changes, the application was responsible for reinitiating streams between the server and the client.

We wanted to realize "reasonable" stream handover from one network to another, meaning that the switch might be audible, but it needed to continue from where it left off. Furthermore, different networks could support different bit rates with different audio qualities: when possible, offer the best quality; otherwise, adjust the quality to the available capacity. We realized this functionality by monitoring the amount of data in the stream buffer and the notifications offered through the NAL. When the streaming buffer size was below a certain threshold, the player didn't consider the currently used link for some time and reestablished the stream over another link (if available). When an NAL notification indicated that the current link wasn't available anymore or that a better link was available, it reestablished the stream over the newly preferred link, without waiting for the buffer to reach the low watermark. This straightforward application logic delivered continuous streams over GPRS and 802.11 and switched between them when the 802.11 network was switched off and on. The switch was audible due to the difference in quality as well as a loss of sync of less than one second. The client used only link and gateway path segment notifications (to get the local IP address to bind to the stream socket) received from the NAL.

Our experiments on power consumption and actual NRM use by an existing streaming player show that it's feasible to offer the network resource abstraction in a realistic mobile device and network configuration, and that this abstraction helps applications straightforwardly adapt to available network resources. To more broadly validate the

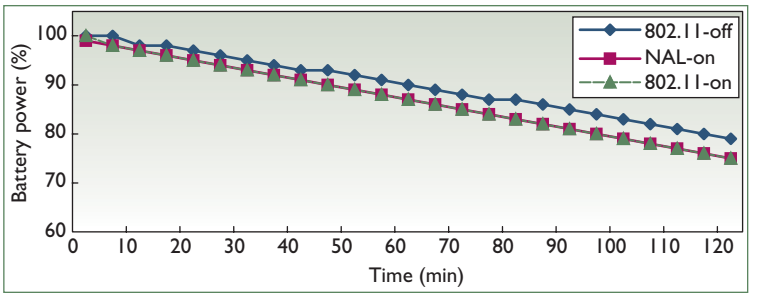


Figure 8. Network abstraction layer (NAL) power usage. We ran a Qtek 9090 Windows CE device with and without the NAL service in combination with the NAL client.

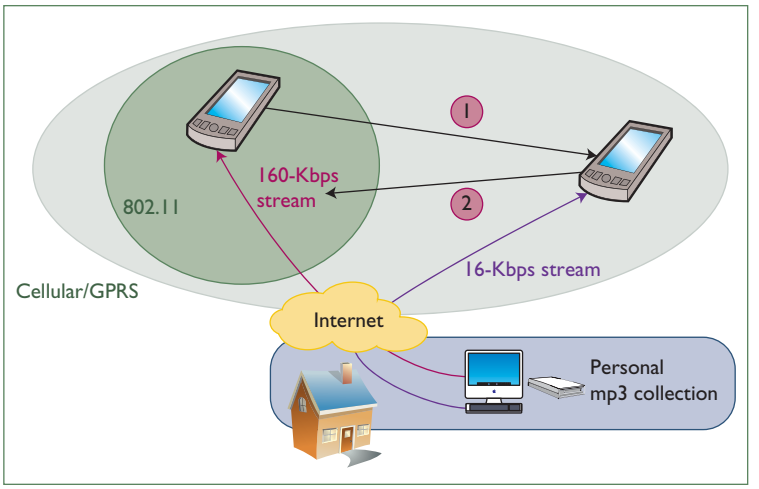


Figure 9. Validation environment. We validated the network resource model (NRM) abstraction using a streaming audio player on a Windows CE device roaming between an 802.11 and a cellular network.

Related Works in Network Resource Usage on Mobile Devices

The network resource model (NRM) and network abstraction layer (NAL) work presented here is based on the principles of providing awareness and control to mobile applications — in terms of network resource usage — and of offering a formal but flexible abstraction that reflects the real state in the layers below the application layer. To the best of our knowledge, no competing approaches exist that follow the same principles we introduce here. However, many aspects of our work relate to other research, notably in the areas of models for cross-layer information interchange, abstractions for network resources, mechanisms for application and network adaptation, and mobility management in heterogeneous network environments.

The API that Marten Bandholz and his colleagues present[1] shows similarities with our NAL interface. It has a modular architecture and defines a query interface for retrieving link-layer information structured as a type hierarchy. It focuses primarily, however, on the link layer — rather than providing a full cross-layer interaction — which makes it harder for applications to take into account relevant network- and transport-layer aspects such as available paths and the state of mobility-management facilities. Furthermore, it doesn't show how different applications' different needs are handled.

Another framework[2] introduces aspects of applications that are aware of the mobility-management process on mobile devices. It defines triggers relevant to a mixed set of mobility-management protocols and shows how applications can interact with these protocols. Our current work has a broader scope because it looks beyond application awareness in relation to mobility-management protocols only.

Gruia-Catalin Roman, Christine Julien, and Qingfeng Huang provide a formal definition of network context for ad hoc mobile nodes.[3] They primarily look into specifying ad hoc network topology within the vicinity of a mobile host and define the cost for paths taken through this topology, but they don't elaborate on other aspects, such as mobility-management facilities.

Many approaches focus on ways mobile applications can adapt to changing resources. The work we present proposes a means of supplying information about network resources so that applications can adapt, but doesn't specify a mechanism for application-layer adaptation itself. The actual strategies for adaptation or mechanisms forcing the application in a certain form to adapt are important, however, because they provide hints on what kind of information is relevant. The Odyssey platform for adaptive mobile data access models the adjustment of applications around the high-level concepts of agility and fidelity. The adaptation functionality is divided between the application and the operating system.

continued on p. 43

abstraction and taxonomy that NRM offers, however, would require us to conduct tests with many more – preferably highly different – applications. We've followed a pragmatic approach by not making strong assumptions on the availability of facilities that support mobility management, such as Mobile IP, yet at the same time have incorporated them into our model in case they are available on the device. The model's extensibility as well as the reference implementation should support adding such facilities. This extensibility also allows the NRM and the NAL to incorporate new network technologies that will most certainly find their way into future mobile devices.

One limitation of our approach is limited type flexibility: currently, developers can't add characteristics to types in a dynamic manner, and the NRM has single inheritance without such constructs as interface inheritance. We could add this, although it would create more complexity and runtime demands. Also, the implementation doesn't have a uniform mechanism for specifying delta values that trigger update notifications to, for instance, notify applications only when a signal strength value is more than a threshold value different from the previous.

The ability to have a continuous and full overview of existing, dynamic network resources on a mobile device opens up exciting new directions for future research. We plan to collect this information from real users as they carry their mobile devices with them every day. This data is likely to show mobility patterns that could help provide additional information to applications that adapt in roaming situations – information that captures users' historical behavior. It could even provide insight on the user-specific correlation between independent technologies' measured characteristics, such as the correlation between visible Bluetooth nodes and in-range 802.11 access points, which can help better adapt or optimize important mobile device resources, such as battery power.

Acknowledgments

The Dutch Freeband Communication Research Programme (Awareness) project supported this research under contract BSIK 03025.

References

1. K. McCloghrie and M. Rose, eds., "Management Informa-

Related Works in Network Resource Usage on Mobile Devices (cont.)

Experience with this system shows that adaptation must strike a balance between agility and stability;[4] for a more continuous service to the user, rapid changes in network resources must not always result in swift adaptation.

Other researchers have also proposed many solutions to separate actual mobility management from application logic, be it at the network layer or the transport layer, to hide most or all roaming aspects for the application. Our work argues against such a separation for certain types of applications. One proposed mechanism,[5] for example, is based on Mobile IP in which the host has multiple home addresses to support the concurrent use of multiple network interfaces. It defines an API to control the mobility-management process, but doesn't provide details on how applications can conveniently use it.

In our definition of the network resource model, a mobile host can provide multiple facilities for mobility management to handle the dynamically changing Internet attachment points. Another work[6] introduces a mechanism that adapts mobility management depending on the application's behavior — that is, the kind of connection the application initiates. The application itself isn't adaptive.

The topics covered by the IETF Detecting Network Access (DNA) workgroup (www.ietf.org/html.charters/dna-charter.html) as well as the IEEE 802.21 standardization effort on handover between heterogeneous networks (www.ieee802.org/21/) look at mechanisms for transfering information from one layer to another. However, these have a narrower scope, focusing instead on interaction between the link and network layers.

References

1. M. Bandholz et al., "Unified Link-Layer API Enabling Wireless-Aware Applications," *Proc. Int'l Symp. Personal, Indoor and Mobile Radio Communications* (PIMRC 06), IEEE CS Press, 2006, pp. 1–5.
2. A. Peddemors, H. Zandbelt, and M. Bargh, "A Mechanism for Host Mobility Management Supporting Application Awareness," *Proc. 2nd Int'l Conf. Mobile Systems, Applications, and Services* (MobiSys 04), ACM Press, 2004, pp. 231–244.
3. G.-C. Roman, C. Julien, and Q. Huang, "Network Abstractions for Context-Aware Mobile Computing," *Proc. 24th Int'l Conf. Software Eng.*, ACM Press, 2002, pp. 363–373.
4. B. Noble and M. Satyanarayanan, "Experience with Adaptive Mobile Applications in Odyssey," *Mobile Networks and Applications*, vol. 4, no. 4, 1999, pp. 245–254.
5. J. Ylitalo et al., "Dynamic Network Interface Selection in Multihomed Mobile Hosts," *Proc. Hawaii Int'l Conf. System Sciences* (HICSS 03), IEEE CS Press, 2003, pp. 315.
6. X. Zhao, C. Castelluccia, and M. Baker, "Flexible Network Support for Mobility," *Proc. 4th Ann. Int'l Conf. Mobile Computing and Networking* (MobiCom 98), ACM Press, 1998, pp. 145–156.

tion Base for Network Management of TCP/IP-Based Internets: MIB-II," IETF RFC 1213, Mar. 1991; http://tools.ietf.org/html/rfc1213.
2. V. Kawadia and P. Kumar, "A Cautionary Perspective on Cross Layer Design," *IEEE Wireless Comm.*, vol. 12, no. 1, 2005, pp. 3–11.
3. A. Peddemors, I. Niemegeer, and H. Eertink, "An Extensible Network Resource Abstraction for Applications on Mobile Devices," *Proc. 2nd Int'l Conf. Communication System Software and Middleware* (COMSWARE 07), IEEE CS Press, 2007.

Arjan Peddemors is a research engineer at Telematica Instituut and a PhD candidate in computer science at the Wireless and Mobile Communication group at Delft University of Technology. His research interests include network communications in the domains of mobile, pervasive, and autonomic computing. Peddemors has an MS in electrical engineering from the University of Twente. Contact him at arjan.peddemors@telin.nl.

Henk Eertink is a research fellow at Telematica Instituut where he leads the Intelligent Communication (INCA) group. His research interests include simplicity and trustworthiness aspects in mobility management and context-aware systems. Henk has a master's and a PhD in computer science from the University of Twente, the Netherlands. He is a member of the IEEE. Contact him at henk.eertink@telin.nl.

Ignas Niemegeers holds the chair of Wireless and Mobile Communications at Delft University of Technology, has been a professor in the computer science and electrical engineering faculties of the University of Twente, and has served as the scientific director of the Centre of Telematics and Information Technology (CTIT), The Netherlands. His present research interests are Beyond 3G wireless infrastructures, UWB networks, ad hoc networks, personal networks, cognitive networks, and ubiquitous computing and communication. Niemegeers has a degree in electrical engineering from the University of Gent, and an MS and a PhD in computer engineering from Purdue University. Contact him at i.niemegeers@ewi.tudelft.nl.

Mortaza Bargh is a member of the scientific staff at Telematica Instituut, where he has been involved in many projects concerning mobile networks, services, and applications. His research interests are in pervasive communication and security, where he currently focuses on mobility management protocols and algorithms for secure and seamless handovers. Bargh has an MS and a PhD in electrical engineering from Eindhoven University of Technology. Contact him at mortaza.bargh@telin.nl.