A Next Generation Operator Environment to Turn Context-aware Services into a Commercial Reality

Oriana Riva[1]*, Veli-Matti Teittinen[2], Sebastian Siikavirta[1], Lasse Huovinen[2]†
[1] Department of Computer Science, University of Helsinki, Finland
[2] Nokia Siemens Networks, Finland
veli-matti.teittinen@nsn.com, siikavirta@cs.helsinki.fi

Abstract

Network operators can play an essential role in turning widely investigated context-aware services into a commercial reality. The Nokia Siemens Networks NEON project aims to develop a virtual network operator environment that can leverage the opportunities offered by packet-based multi-access networks, increasingly more powerful mobile devices, and service provider business practices to enhance the end user experience with value-added services. This paper primarily focuses on the support provided by the NEON network to fully accomplish context-aware services. We present the design and implementation of the NEON network currently in use in Nokia Siemens Networks premises with a special focus on the context services it provides. We assessed the performance of such services in terms of latency and power consumption on smart phones and we used them to implement the Communication Dispatcher, a prototype application that enables efficient context-aware user-to-user multi-terminal communication.

1 Introduction

Context-aware services that dynamically adapt to the users' environment and needs (i.e., context) have been investigated intensively in the last 20 years. Services of this type have appeared in many domains such as tourism, shopping, entertainment, driving assistance, and healthcare. Deployed services have usually relied on a customized infrastructure that supports collection, processing, and distribution of context information for each specific service and in each specific domain. However, despite the potential benefits context-aware services could bring to the community, so far they have not been able to leave their academic laboratories and turn into large-scale commercial realities.

A main reason for this unsuccessful outcome is the lack of a common infrastructure that can make context-aware services widely deployable. Network operators, driven by the need to attract as many customers as possible, while containing operational costs, have finally recognized the benefits achievable by providing such a common infrastructure. This comes with almost no added cost because information such as the location of users and the status of the network connectivity is naturally available to network operators. Yet, this kind of information is today almost completely unused. New business models arise and competition increases among network providers, device vendors, and service providers. Therefore, the challenge for network operators is to identify a minimal set of common services that can allow application developers to deploy innovative context-based services and quickly accommodate users' future needs.

In the last two years, the Nokia Siemens Networks NEON (Next generation operator environment) project has investigated how to effectively take advantage of the opportunities offered by packet-based multi-access networks, ever-increasing capabilities of near-ubiquitous mobile devices, widespread services offered by many service providers such as Google or Yahoo, and emerging interaction models (e.g., peer-to-peer) that go beyond classical consumer-provider relationships. The core idea of the NEON project is to build a service-rich experimental environment through an overlay network that can enhance the end user experience over heterogenous multi-access networks. Specifically, the NEON project developed an overlay network environment providing location support, VoIP, instant messaging, device management, and identity management with the aim of experimenting with virtual network operators in enterprise settings. Currently, the NEON network runs in Nokia Siemens Networks buildings (in the Helsinki area) and is accessible from the Internet [9]. At the moment, it hosts more than 200 registered users.

*Her new affiliation is ETH Zürich, Department of Computer Science. Contact her at oriva@inf.ethz.ch
†His new affiliation is EADS Secure Networks. Contact him at lasse.huovinen@hut.fi

978-0-7695-3154-0/08 $25.00 © 2008 IEEE
DOI 10.1109/MDM.2008.19

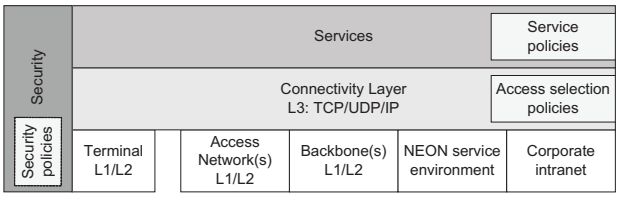

Figure 1. NEON layered reference model

In this paper, we focus on the context-awareness support the NEON network provides and the types of context-aware services it enables. The rest of the paper is organized as follows. The NEON architecture and its context services are introduced in Section 2 and Section 3 respectively. Section 4 presents results of our experimental evaluation while the Communication Dispatcher prototype using the NEON context services is described in Section 5. Section 6 discusses related work and the paper concludes in Section 7.

2 NEON: A Next Generation Virtual Network Operator

NEON (Next Generation Operator Environment) is an experimental overlay network that has been built with the aim of assessing new technological opportunities, investigating non-traditional interaction models, and demonstrating the feasibility of novel value-added services, such as context-aware services. The basic idea behind NEON is to build a virtual network operator (VNO) environment capable of providing IP-based services across multiple access networks. By hiding the underlying network complexity and offering modular service packages, a VNO permits simplifying service development and lets application developers focus on the user needs. This section presents the NEON architecture and gives insights into its implementation.

2.1 Reference Model

Figure 1 shows the three-layer reference model of the NEON infrastructure. The bottom layer includes access and backbone networks providing physical and transport network, intranets, and NEON-specific service environments. In principle, the NEON infrastructure can work on top of any current or future access network that supports IP.

The middle layer is the connectivity layer that is responsible for providing end-to-end connectivity between nodes in the NEON infrastructure using IP at the network layer and UDP/TCP at the transport layer. Nodes may be mobile as well as fixed devices. The connectivity layer provides support for tunnelling and secure communication when necessary. The highest layer is the service layer consisting of the actual value-added NEON services.

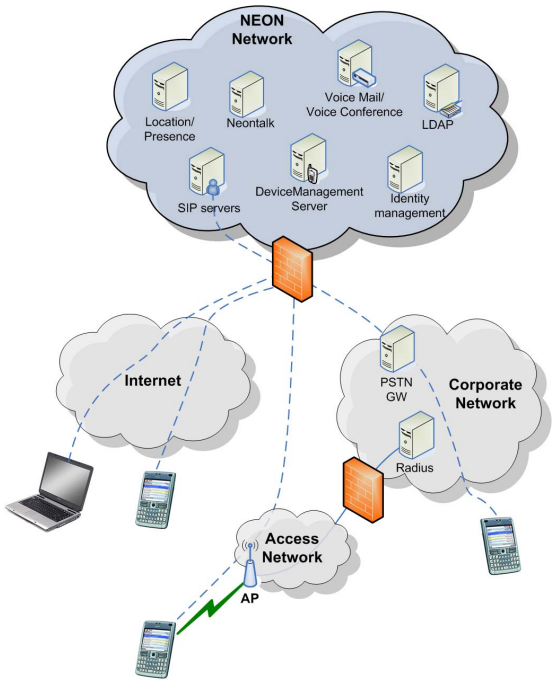

Figure 2. NEON network architecture

Security spawns all three system layers although in NEON the focus is on security issues at the connectivity and service layer. Service-specific policies define how the service should be configured, security policies regulates network access, authentication, and traffic encryption, and access selection policies specify how access networks should be selected for supporting specific service communications.

2.2 Architecture and Implementation

Figure 2 shows the NEON network architecture. A terminal can access NEON services, using WLAN, cellular networks, or any other access network. The connection between an access network and the NEON network can be a point-to-point link or can go through one or multiple intermediate networks as well as the Internet.

Services are hosted by the NEON virtual overlay network. Core NEON services include SIP-based voice over IP, XMPP-based instant messaging (using the Neontalk server), voice mail and voice conference, and location and presence services. The level of security and authorization can vary depending on the situation. NEON integrates identity management and LDAP (Lightweight Directory Access Protocol) [15] servers to provide secure authentication, authorization, and accounting. This also guarantees secure access to WLAN, cellular, and other access networks as well

as to services hosted by NEON and corporate networks. Credentials of different types, for example based on SIM cards, must be associated to each terminal, service, and network "security point" (responsible for carrying out authentication, authorization, and encryption).

To bridge conventional telephone networks and VoIP SIP infrastructures, the NEON network is connected to the PSTN network through a PSTN Gateway. The RADIUS (Remote Authentication Dial In User Service) [13] server is used for WLAN authentication.

The NEON network has been implemented on top of the Nokia Siemens Networks infrastructure and uses one of the WLAN networks available in the campuses in the Helsinki area, PSTN Gateways, and another corporate network and its services. NEON services are accessible over the Internet from the NEON service portal [9]. Currently, the access is limited to Nokia and Nokia Siemens Networks internal users. At the moment, NEON hosts a live network with more than 200 registered users.

The NEON implementation consists of various building blocks: open source, product, and in-house implementations. The registrar SIP server is an open source solution complemented with in-house media gateway and SIP proxy implementations. Identity management, XMPP, and LDAP as well as Asterisk-based Voice Mail and Voice Conference are open source implementations. In addition, NEON provides services that are pure in-house implementations, such as location and presence services.

To access NEON, users need to have credentials that are created through the NEON portal. Furthermore, to improve the usability of the NEON services, in addition to traditional identity management (i.e., user-password login procedure), NEON also provides advanced identity management by utilizing Single Sign-On implementation. For mobile devices, NEON offers provision capability for installing VoIP and access network settings by utilizing Nokia Device Management, a Nokia product following OMA Device Management [11] standards.

3 Context-awareness Support in NEON

NEON currently supports provisioning of (indoor and outdoor) location, presence, user status, calendar, and terminal information by means of the ContactsNow application (see Figure 3). This is an application written in Python that runs on Symbian mobile phones.

To support indoor positioning, the application scans WLAN Access Points (APs) as well as current serving GSM or WCDMA cells. Radio measurements are then periodically delivered as XML data over HTTP to the NEON location server. By knowing the locations of the APs and base stations, the server calculates the location of the device and returns name of the building, floor, and wing information,

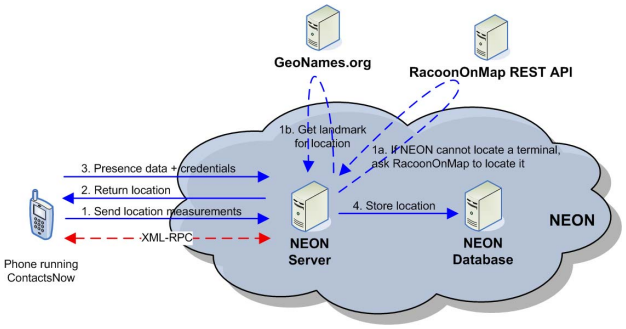

Figure 3. NEON ContactsNow execution

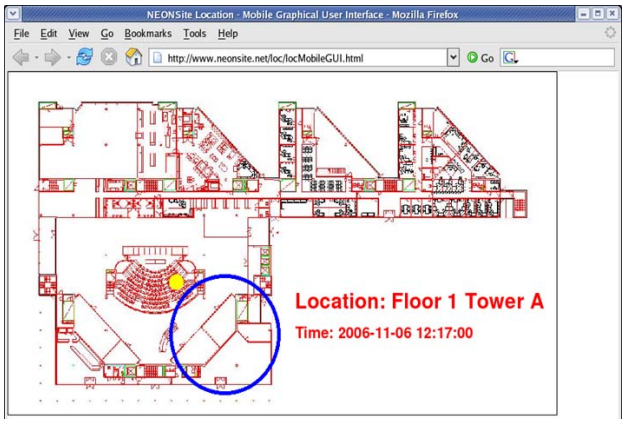

Figure 4. NEON indoor user localization

as depicted in Figure 4. For outdoor positioning, ContactsNow employs a GPS receiver integrated in the device or connected via Bluetooth. In addition, if the user allows, the server stores all location data in a location database. The location server is an XML-RPC server providing location computation and storage as RPC functions.

If for any reason, the NEON location server is not capable of calculating the position of the terminal based on WLAN measurements, it can request the RaccoonOnMap server to calculate the position. The RaccoonOnMap server is the same as the one used by the RaccoonOnMap portal [12]. The location server sends the serving cell information received from the terminal to the RaccoonOnMap server via RESTful interface. The RaccoonOnMap server returns the corresponding coordinates which are eventually delivered to the terminal. Furthermore, to convert location measurements in higher-level landmarks the NEON location server can interact with the GeoNames service [6].

In addition to location information, ContactsNow collects other user and terminal information such as user availability (free or busy by checking the current calendar en-

tries), device profile (normal, silent, meeting, etc), terminal status (active or idle, for instance, by checking when any keyboard key was last time pressed), terminal identification (IMEI code), and terminal capabilities. Uniquely identifying a user's terminal is essential to support multi-terminal interactions in which a user has more than one terminal active at a time. If the user allows, also this data can be stored in the NEON databases, and the user can also decide the frequency of data collection. Essentially, anyone can ask NEON to calculate his location given some radio measurements but only users having a NEON account are allowed to store data to and fetch data from the NEON databases.

In collecting context information, a distinction needs to be made between static and dynamic information as well as proactive and reactive monitoring. Dynamic information needs to be more frequently collected and published in order to guarantee good accuracy. A proactive approach in which context monitoring is constantly performed also contributes to achieve high accuracy. On the other hand, high frequency and proactive collection yield higher costs in terms of network bandwidth utilization and power consumption.

4 Experimental Evaluation

The aim of the experimental evaluation was to assess the performance of the NEON context services. We measured the latency of the ContactsNow application in monitoring and storing location and presence information, and the latency of the NEON location server in processing location data. We also quantified the cost of ContactsNow in terms of power consumption on smart phones. Experiments were run in the NEON live network built in Nokia Siemens Networks premises. Notice that by being a office environment, there exists background noise due to other mobile phones, WLANs, Bluetooth, etc. All NEON services run on Linux machines and Nokia Series N and E phones. A Nokia N95 phone was used in the measurements.

As reported in Table 1, in the latency experiments we distinguish three cases:

1. WLAN/WLAN: ContactsNow communicates with the NEON server using WLAN and the location computation uses WLAN APs.
2. WLAN/cell: ContactsNow communicates with the NEON server using WLAN but the location computation uses the cellular network; these tests are run in an unmapped WLAN coverage area.
3. cell/WLAN: ContactsNow communicates with the server using the cellular network and the location is computed using WLAN APs; in these tests, none of the available WLAN networks (SSIDs) was configured on the terminal, such that although the terminal could see them, they were not used for communication.

In the WLAN/WLAN and cell/WLAN case, the location calculation latency as observed by the client (Client Loc Calc Lat) includes the time to encode XML data, the time to send the location calculation request (roughly 4.2 kB) from the client to the server, the processing time on the server side, the time to send the location calculation response (roughly 1.5 kB) from the server to the client, and the time to decode XML data. The total processing time on the server side (Calc Total) when WLAN APs are used includes the time to fetch WLAN APs data from the location database, the time to compute the location (WLAN Calc), and the time to decode/encode XML data. The server processing time is minimal (approximately 63 ms) and therefore the total location calculation latency observed by the user is mostly due to the communication latency. Indeed, in the Cell/WLAN case, the observed client latency is higher due to the higher delay of WCDMA networks compared to WLAN networks.

In the WLAN/Cell case, the location is computed using the cellular network. The location computation latency seen by the client is roughly 735 ms. On the server side, the calculation latency accounts for approximately 257 ms. As the number of mapped APs is insufficient, the location computation based on APs fails. Therefore, the location server uses the cellular network for retrieving its serving cell (Cell Calc). Furthermore, the server communicates with the RaccoonOnMap portal (RaccoonOnMap Lat) to get the location of the serving cell the phone is using and fetches the corresponding landmark from GeoNames (GeoNames Lat).

The latency of storing client's presence and location data (Client Pres/Loc Stor) accounts for request transmission, authentication, XML data decoding, actual MySQL storing operations, and response transfer. The authentication time is roughly 11-13 msec. XML data decoding and MySQL operations cost roughly 27-32 msec. The rest is communication latency. The higher latency for the Cell/WLAN case is due to the higher delay of WCDMA networks.

Power consumption was measured using the Nokia Energy Profiler [10] tool. This tool gave us a rough estimation of the power consumed by ContactsNow. In the idle case, when the phone has Bluetooth in inquiry-scan mode, the GSM radio switched on, the back-light and the screen switched on, and the power monitoring tool running, we observed an average power consumption of 0.34 W. When the back-light is turned off the average power consumption is approximately 0.14 W and if also the screen is turned off it drops to 0.09 W. The cost of ContactsNow when idle showed to be negligible.

In the WLAN/WLAN case, ContactsNow sends WLAN measurements to the NEON location server and retrieves the terminal location. As shown in the left side of Figure 5, the power cost of such communications comes in peaks ranging from 1.3 up to 1.8 W. In the cell/WLAN case,

Table 1. ContactsNow latencies (in ms)

Case	Client Loc Calc Lat	Server Loc Calc					Client Pres/Loc Stor
		WLAN Calc	Cell Calc	Calc Total	RaccoonOnMap Lat	GeoNames Lat	
WLAN/WLAN	848.21	53.81	-	63.03	-	-	317.53
WLAN/Cell	734.80	23.92	225.45	256.64	59.41	165.30	296.42
Cell/WLAN	3432.46	54.24	-	63.51	-	-	913.28

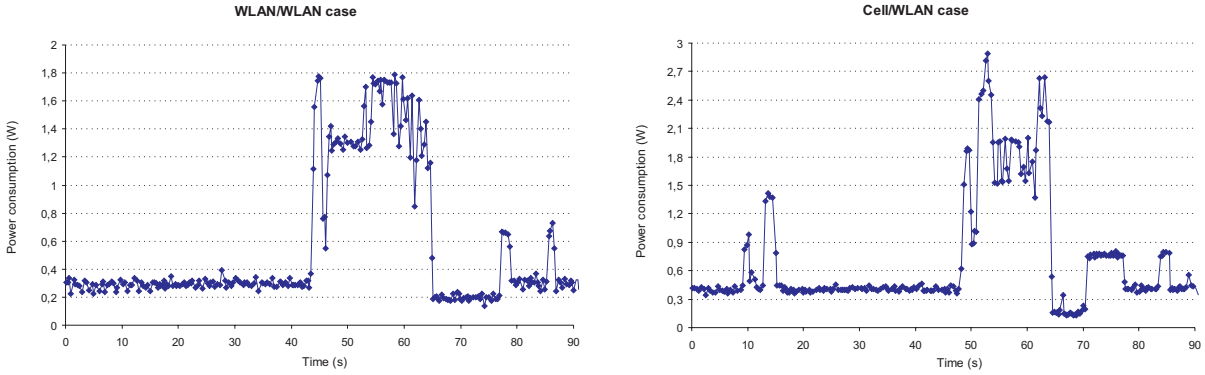

Figure 5. ContactsNow power consumption in the WLAN/WLAN and cell/WLAN case

ContactsNow uses WCDMA for communicating with the location server and, as expected, the consumption is higher than with WLAN. Power consumption peaks reach 2.9 W.

From these results we conclude that the latency and power consumption depend strongly on the type of communication used and on the frequency at which presence/location information is computed and synchronized with the server. The computation that ContactsNow requires on the terminal is relatively constrained.

5 Communication Dispatcher

To demonstrate how the NEON infrastructure can actually be used in building value-added context-aware services we implemented the *Communication Dispatcher* application. The Communication Dispatcher aims to control and optimize the management of incoming and outgoing communications on a user's terminals while minimizing the user involvement. Specifically, the Dispatcher system is capable of identifying the most suitable communication mechanism to be used in contacting a certain party or in being contacted by a remote party. For example, the Dispatcher can recommend to the caller to send a short message or an e-mail if the called party is currently unable to take a phone call.

The "best" way to contact a user depends on several user's contextual information such as current activity, calendar, location, status, nearby people, and available devices. By taking into account this kind of information, the Dispatcher can enhance the users' reachability and at the same time optimize the effectiveness of the communication by avoiding, for instance, plenty of missed calls to busy users. Ultimately, the Dispatcher provides an efficient and unobtrusive way to support communication among people.

5.1 Scenario

To illustrate the usefulness of the Communication Dispatcher we first provide a storyboard.

After her morning coffee and newspaper session with her colleagues, Anne returns to her office and begins to write a deliverable for the research project she is leading. She needs to finish it by noon so she would like to turn off her mobile phone and not to be disturbed, but her 2-year old kid is today sick and she wants to be reachable by her mother who is looking after him at home. On the other side of the building, Jukka is booking his trip to London and needs to get the travel plan approved by his leader Anne before she leaves the office. He would like to phone her, but his Communication Dispatcher can see that Anne is currently busy and therefore recommends to send her an e-mail. Jukka types his message and once Anne has completed her deliverable, she finds Jukka's e-mail and replies to him.

At 6 pm, Anne has a video conference with Boston to discuss a project proposal. In the meantime, her brother is on the bus heading towards home and remembers to have forgotten to invite her sister to spend the week-end at his summer cottage. His Communication Dispatcher checks Anne's status and since Anne is in a meeting, it proposes to him to

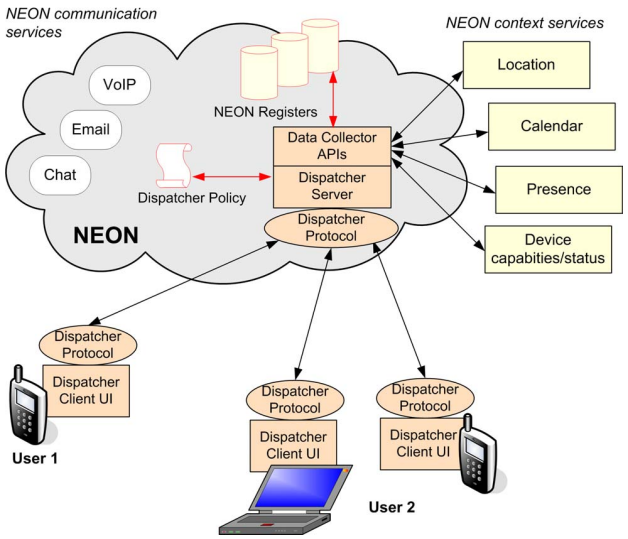

Figure 6. NEON Communication Dispatcher

send her an sms. After the meeting, on the bus back home, Anne reads her brother's sms and accepts his invitation.

5.2 System Design

In the NEON Communication Dispatcher architecture, depicted in Figure 6, NEON provides the Dispatcher with several context services to monitor location, presence information, calendar entries, and status and capabilities of available devices. Furthermore, NEON supports communication mechanisms such as VoIP, e-mail, and chat.

The Dispatcher Server is responsible for identifying the most adequate communication methods for contacting a certain user. It retrieves information about registered users from the NEON Registers and then instructs the NEON context services to collect their context information. To this purpose, the Dispatcher Server has available different types of Data Collector APIs. The execution of the Dispatcher Server is regulated by a Dispatcher Policy specified by system administrators. For example, by dynamically varying the policy specifications, higher privacy and security controls can be enforced at some point to protect disclosure of users' private data in unknown network domains. Or different matching strategies can be used by the Dispatcher Server depending on the users' data availability over time.

Each user terminal runs a Dispatcher Client that supports the communication protocol enforced by the Dispatcher Server and provides the user a UI for modifying her profile information, submitting requests, and retrieving results.

In accomplishing its task, the Dispatcher Server utilizes the following types of information: (i) communication capabilities supported by the caller's terminal and by the terminal to be contacted (e.g., sms, chat, VoIP); (ii) location and presence information of the user to be contacted (e.g., user busy or available, in a meeting or in the office, etc.); (iii) current status of the terminals owned by the user to be contacted (e.g., active or idle terminal); and (iv) communication preferences of the user to be contacted.

Currently, the application considers the following communication mechanisms: voice call, SMS, chat (provided by the NEON infrastructure), VoIP (provided by the NEON infrastructure), e-mail service (provided by the NEON infrastructure), and person-to-person meeting.

In a common case of interaction, the Dispatcher Client requests the Dispatcher Server to contact a certain user identified by a username. The Dispatcher Server verifies if such a user has an account in the NEON system, and if she is currently logged in and with which terminal(s).

In general, each user will be logged into the NEON infrastructure with multiple terminals, each offering different communication capabilities. At any point in time, a terminal may be active (i.e., it has been used recently) or idle. For example, when going to a seminar a user can bring her phone and leave her laptop in the office. The phone will be active and the laptop idle. Therefore, for the entire duration of the seminar, the user would like to be reached by phone only for emergency situations, but the laptop should be preferred for ordinary matters. However, notice that the status of a terminal is generally different from the status of its user. A user in charge of giving a presentation at the seminar will bring both her phone and her laptop. Both devices will be active, but the user will be busy. Finally, a third case is that of a user who comes to the seminar to listen only to a couple of presentations of interest. She brings her laptop to continue working in background (terminal active) and receive her e-mail regularly (user available), while waiting for presentations of major interest to start.

To support multi-terminal scenarios of this type, the Dispatcher Server first matches the communication capabilities supported by the terminal of the caller and by the terminal(s) of the user to be contacted. Then it uses information such as user's location, calendar, and presence information as well as time of the day to infer the user's status and in particular her availability (or non availability) to support a certain type of communication in the current situation. After this, the status of the terminal is used to further filter as well as reorder the list of available communication methods.

Finally, each user is characterized by a profile that among personal profile information contains communication preferences. Communication preferences are specified in such a way that for each terminal registered with the NEON network the user can express a preference on each supported communication capability and associate it to a specific location or activity. For example, when at home the user may wish to be contacted on her phone by sms and when at work

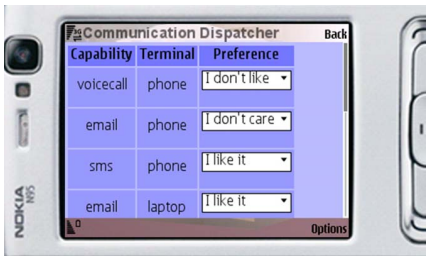

Figure 7. The Dispatcher application permits specifying communication preferences

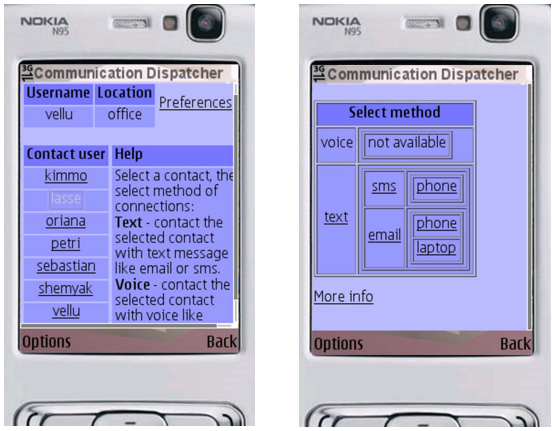

Figure 8. The user specifies a contact (on the left) and the Dispatcher returns the "best" communication methods (on the right)

on her laptop by e-mail. Communication preferences are used to reorder the list of selected communication methods to be returned to the Dispatcher Client. Depending on the user's privacy settings, together with the list of communication methods, the Dispatcher Server can return also several terminal information such that the caller can more accurately choose the most convenient communication method.

5.3 Prototype Application

The NEON Dispatcher Client and Dispatcher Server modules have been implemented in PHP. XML-RPC is used to enable client and server modules to talk to each other. The development was done using laptops running Linux and Nokia N95 phones running SymbianOS.

The Dispatcher Client allows a user to express her preferences on how she would like to be contacted in different locations or activities. Preferences are of three types: "I like", "I don't care", "I don't like". In Figure 7, the user specifies that on her phone she prefers to receive short messages rather than voicecalls while on her laptop she prefers e-mails. These preferences are then associated to the situation "at the office" (not shown in the figure).

In the screenshots presented in Figure 8, if a remote user wishes to contact a user with such preferences, she can select the corresponding username from the list of available contacts or type it. The Dispatcher Client will then retrieve the list of available communication methods for contacting such a user. In this case, the Dispatcher recommends to send a "text" message in the form of sms or e-mail to the user's phone or send an e-mail to her laptop. Depending on the privacy settings of the user to be contacted, it is also possible to retrieve more detailed information on the status of her terminals (e.g., "laptop idle since 2 hours").

6 Related work

Mobile Virtual Network Operator (MVNO) is a phenomenon in which an operator or a company provides mobile phone services without having its own frequency allocation of the radio spectrum and without owning its own network infrastructure. A MVNO has contracts to use the existing network(s) of other providers, and can independently fix its own pricing and offer different value-added services. Currently, there are roughly 360 planned or operational MVNOs world-wide. For example, Virgin Mobile [2] is the world's first MVNO, launched in the United Kingdom in 1999, and now with more than four million customers. Blyk [1] is an advertising-based MVNO offering free phone calls and messaging in return for sending customers advertising. Mobile operators benefit from MVNOs because MVNOs can attract more customers by addressing specialized service requirements of certain customer niches with customized applications, price tariffs, marketing, etc. Phenomenon such as MVNO demonstrates how the operator environment is undergoing fundamental changes due to the pressure for increased profitability.

Partly from a business perspective, but more importantly from a technical perspective, the proliferation of multi-access networks and increasingly more powerful mobile devices as well as the emergence of new user interaction models motivate the NEON research project. NEON aims to provide an experimental overlay network integrating innovative platforms where new technologies and more advanced services can be experimented. The core approach of NEON is to build a service-rich virtual operator environment as an overlay network across multi-access networks.

NEON specifically aims to accomplish context-aware services and address the demands of mobile users for a continuous access to personalized adaptive services. In the last 20 years, a plethora of research projects have actively investigated these issues in research areas such as

ubiquitous computing, mobile computing, augmented reality, and human-computer interaction. Researchers have focused on supporting context provisioning through modular infrastructures and middleware architectures [4, 14]. Especially, several positioning mechanisms for indoor and outdoor location monitoring have been devised [7]. Research work on context reasoning, context prediction, and ontologies has provided algorithms and methods to represent high-level context information and dynamically adapt the application's behaviour to environmental changes [8]. However, although many examples of context-aware system prototypes exist such as tourist guides, fieldwork applications, and healthcare applications, context-aware services have not managed to leave their research laboratories and step out to the real-world. NEON specifically targets the actual deployment of context-aware services on a very large-scale by means of network operator environments that can more efficiently collect and distribute several types of context information.

The Communication Dispatcher application share ideas with other research projects that have focused on providing automatic and personalized access network selection. For example, in [5], the authors proposed an intelligent software agent that transparently and constantly selects the most suitable network service (in terms of network QoS and connectivity) based on the user's profile. Or the concept of "Personal Broadband" [3] that has been proposed in the context of the Communications Futures Program (CFP) at MIT defines a set of capabilities and interfaces that allow users to select the connections that best meet their needs within a particular context (e.g., coverage, cost, bandwidth, quality of service guarantees). Our Communication Dispatcher implements a use case that could motivate any of these research projects and demonstrates how services of this type can be achieved through the NEON's context services.

7 Conclusions and Future Work

This paper presented the NEON approach to supporting context-aware services. NEON provides a virtual network operator environment capable of collecting user context information, such as location, presence, and calendar entries, and of supporting different types of communication across heterogenous multi-access IP-based networks. We combined the NEON context and communication services to build a proof-of-concept application called Communication Dispatcher that enables efficient user-to-user multi-terminal communication. The NEON approach demonstrated to be a viable solution to fully accomplish quick and efficient deployment of context-aware applications. The NEON network already hosts 200 registered users. In the near future, we plan to carry out extensive user evaluation by installing applications such as the Communication Dispatcher on users' personal mobile devices. Furthermore, we will continue to develop the NEON testbed to offer a service-rich environment for different R&D purposes.

Acknowledgments

The authors would like to thank Hannu Flinck, Kimmo Raatikainen, Petri Velin, and all NEON project team.

References

[1] Blyk. http://www.blyk.co.uk, 2007.
[2] Virgin Mobile. http://www.virginmobile.com/vm, 2007.
[3] Broadband Working Group MIT Communications Futures Program (CFP). Vision of Personal Broadband. White paper, January 2006.
[4] A. K. Dey, D. Salber, and G. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2-4):97–166, 2001.
[5] P. F. G. Lee, S. Bauer, and J. Wroclawski. A User-Guided Cognitive Agent for Network Service Selection in Pervasive Computing Environments. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom'04)*, pages 219–228. IEEE Computer Society, March 14-17 2004.
[6] GeoNames. http://www.geonames.org/, 2007.
[7] J. Hightower and G. Borriello. A survey and taxonomy of location sensing systems for ubiquitous computing. Technical Report UW CSE 01-08-03, University of Washington, Department of Computer Science and Engineering, August 2001.
[8] P. Korpipaa, J. Mäntyjärvi, J. Kela, H. Keranen, and E. Malm. Managing Context Information in Mobile Devices. *IEEE Pervasive Computing*, 2:42–51, July-Sept 2003. IEEE Educational Activities Department.
[9] NEON portal. http://www.neonsite.net, 2007.
[10] Nokia Energy Profiler. http://www.forum.nokia.com/main/resources/development_process/power_management/nokia_energy_profiler, 2007.
[11] OMA (Open Mobile Alliance) Device Management Working Group. http://www.openmobilealliance.org/tech/wg_committees/dm.html, 2007.
[12] RaccoonOnMap. http://raccoon.openlaboratory.net/RaccoonOnMap, 2007.
[13] C. Rigney, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, Internet Engineering Task Force, June 2000. Available at http://www.ietf.org/rfc/rfc2865.txt.
[14] O. Riva. Contory: A Middleware for the Provisioning of Context Information on Smart Phones. In *Proceedings of the 7th ACM International Middleware Conference (Middleware'06)*, volume 4290 of *Lecture Notes in Computer Science*, pages 219–239. Springer-Verlag, 2006.
[15] J. Sermersheim. Lightweight Directory Access Protocol (LDAP). RFC 4511, Internet Engineering Task Force, June 2006. Available at http://www.ietf.org/rfc/rfc4511.txt.