

CAR-2-X Communication SDK – A Software Toolkit for Rapid Application Development and Experimentations

Andreas Festag, Roberto Baldessari, Wenhui Zhang and Long Le
NEC Europe Ltd.
NEC Laboratories Europe
Kurfürstenanlage 36, D-69115 Heidelberg, Germany
Email: {festag|baldessari|zhang|le}@nw.neclab.eu

Abstract—The *CAR-2-X Communication SDK* is a software implementation of a networking protocol stack for car-to-car and car-to-infrastructure communication with IEEE 802.11 radio technology. The core of the SDK is a routing protocol that provides wireless ad hoc and multi-hop communication using geographical positions for addressing and packet forwarding. The routing protocol scheme is enhanced by various features to meet the requirements of safety and infotainment applications in vehicular environments.

The SDK can be executed in various hardware environments, such as virtual machines, software development systems and embedded hardware for automotive and road side units. The SDK enables rapid development of safety and infotainment applications for CAR-2-X communication by providing state-of-the-art communication protocols and well-defined application programming interfaces (APIs). The advanced features and a high grade of configurability makes the SDK an ideal software platform for experimentation and evaluation of CAR-2-X communication systems in laboratory environments and real world testbeds.

In this paper we describe functionalities and features of the *CAR-2-X Communication SDK* and present the SDK's implementation aspects for the protocols and API. Then, we present a framework for benchmarking and evaluation of CAR-2-X communication protocols. The *CAR-2-X Communication SDK* is available under the conditions of a software license and provided royalty-free [1].¹

I. INTRODUCTION

Car-to-car and car-to-infrastructure communication (also known as CAR-2-X communication) has gained significant interest in the last years. It is regarded as a promising technology to increase road safety, to improve road traffic efficiency and to provide comfort applications for drivers and passengers. In Europe, various R&D projects (such as *FleetNet*, *WillWarn*, *NoW-Network on Wheels*, *CVIS*, *SafeSpot*, *SeVeCoM*, *GEONET* and *COMeSafety* [2]–[9]) have consolidated a solid technical basis for a communication system based on IEEE 802.11 radio and ad hoc networking. The efforts have led to the design of communication paradigms, mechanisms and protocols that are highly optimized to meet the specific application requirements and communication environments. Recently, standardization of the system has been

initiated [10] and the first field operational tests (FOTs) are under preparation (e.g. PRE-DRIVE C2X [11]) or have started (SIM-TD).

The *CAR-2-X Communication SDK* provides a software implementation of the protocol stack for on-board units (OBUs) in cars and roadside units (RSUs). The implementation comprises basic networking functions and many advanced protocol features that are relevant for the communication in real environments and are foreseen as required functions in future systems. The core concept is wireless, ad hoc communication utilizing geographical positions for addressing and routing of data packets (Geo-networking). Advanced features include multi-hop communication, geographical addressing of individual nodes and areas defined by geometric shapes, mechanisms for improving the reliability and efficiency of data transmissions, data security, privacy and the integration of Internet protocols.

Originally, the SDK was implemented as a proof-of-concept demonstrator and has evolved to a reference implementation for CAR-2-X communication protocols. The SDK is in-line with the efforts of the *CAR-2-CAR Communication Consortium* [12] and is used by several European R&D projects and demonstrations [13] as communication platform.

The SDK can be regarded as a research-oriented, experimental software platform that bridges the time until standardization efforts have completed and commercial products appear in the market. The SDK addresses two main groups of potential users: First, application developers can utilize the SDK's API implementation for rapid software development and rely on the SDK as the underlying protocol stack. Second, system engineers can make use of the SDK in order to conduct experiments with CAR-2-X communication, test specific configurations and assess the performance of particular features. Such experimental measurements, as compared to other techniques, i.e., analysis and simulations, can give more accurate results. Finally, the SDK gives feedback on various aspects of the system, helps improving protocol design, and potentially accelerates development of the overall system.

This paper presents the *CAR-2-X Communication SDK* as a toolkit for developing safety, traffic efficiency and infotainment applications and for benchmarking of CAR-2-X communication protocols. The next section gives some technical back-

¹A. Festag was partially supported by the EU FP7 project PRE-DRIVE C2X, grant number 224019. L. Le was partially supported by the EU FP7 project INTERSAFE-II, grant number 223951.

ground on Geo-networking as the core networking concept of the SDK. Then, Sec. III details implemented functionalities and features and Sec. IV describes implementation design, configuration and usage of the SDK. Sec. V presents a framework for performance evaluation of CAR-2-X communication protocols, and Sec. VI concludes the paper.

II. TECHNICAL BACKGROUND ON GEO-NETWORKING

The core networking concept used in the SDK is *Geo-Networking*, an ad hoc routing protocol that provides wireless multi-hop communication over short-range wireless radio without the need of a coordinating infrastructure as in cellular networks. The basic idea of Geo-networking had originally been proposed for mobile ad hoc networks, e.g. [14], and has been optimized for various systems, such as wireless sensor networks and vehicular ad hoc networks.

In principle, Geo-networking provides data dissemination in ad hoc networks by using geographical positions of nodes. For multi-hop communication, nodes forward data packets on behalf of each other. Geo-networking is attractive in vehicular environments for two reasons. First, it works well in highly mobile networks where network topology changes frequently. Second, Geo-networking offers flexible support for heterogeneous application requirements, including applications for road safety, traffic efficiency and infotainment. In particular, Geo-networking provides periodic transmission of safety status messages at high rate, rapid multi-hop dissemination of packets in geographical regions for emergency warnings, and unicast packet transport for infotainment applications.

Geo-networking provides two basic and strongly coupled functions: *geographical addressing* and *geographical forwarding*. With geographical addressing, Geo-networking can address a node by its position or address multiple nodes in a geographical region (geo-address). For geographical forwarding, Geo-networking assumes that every node has a partial view of the network topology in its neighborhood and that every packet carries a geo-address, i.e., the geographical position or geographical area as the destination. When a node receives a data packet, it uses the geo-address in the data packet and its own view of the network topology to make an autonomous forwarding decision. Thus, packets are forwarded 'on the fly' and nodes do not need to set up and maintain routes. Geo-networking assumes that every network node knows its geographical position, e.g. by GPS or another positioning systems, and maintains a *location table* of geographical positions of other nodes as soft states.

Geo-networking has three core protocol components: beaconing, location service and forwarding. *Beacons*, also referred to as *heartbeats*, are small packets that each node broadcasts periodically to inform other nodes about its ID, its current geographical position, its speed, and its heading. Nodes can cooperatively provide a *location service* that resolves a node's ID to its current position on a query/reply basis.

Forwarding basically means relaying a packet towards the destination. There are three types of Geo-networking forwarding: *Geographical Unicast (GeoUnicast)* (Fig. 2a) offers

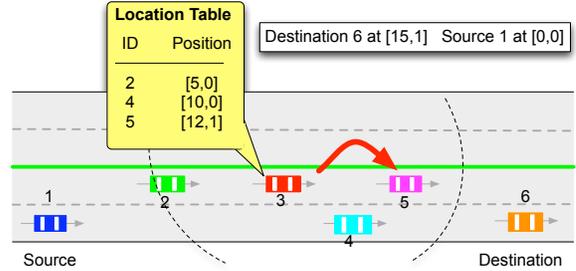


Fig. 1. Principle of Greedy Forwarding: Node 3 Selects 5 as Next Hop

packet relay between two nodes via one or multiple wireless hops. When a node wants to send a unicast packet, it first determines the destination position (by means of location table look-up or the location service) and forwards the data packet to the next node in the direction of the destination. This node in turn re-forwards the packet along the path until the packet reaches the destination. Greedy forwarding [14] is an algorithm for unicast, where a node selects the next hop based on position information such that the packet is forwarded in the geographical direction of the destination (Fig. 1). *Geographical Broadcast (GeoBroadcast)* (Fig. 2b) distributes data packets by flooding, where nodes re-broadcast a packet if they are located in the geographical region specified by the packet. Packet is forwarded only once: If a node receives a duplicate packet that it has already received, it will drop the packet. *GeoAnycast* is similar to *GeoBroadcast* but addresses any node in a geographical area. *Topologically-Scoped Broadcast (TSB)* (Fig. 2c) offers re-broadcasting of a data packet from a source to all nodes that can be reached in certain number of hops (all nodes in a n-hop neighborhood). Single-hop broadcast is a special case of TSB that is used to send *heartbeats* including application data payload.

In Geo-networking, a node usually processes all data packets that it receives on the wireless links² to keep track of all nodes in its surrounding. Since each data packet contains the source's and previous forwarder's positions, a receiving node can update its location table accordingly. A Geo-networking packet header contains fields for node identifier, position and timestamp for source, sender, and destination, and more.³ Fields in a Geo-networking header are classified as immutable and mutable: *Immutable* fields are not changed in the forwarding process. On the other hand, *mutable fields* are updated by forwarders. This allows a forwarder to change some header fields on the fly, e.g., in case it has more recent information in its location table about a given destination.

III. FUNCTIONALITIES AND FEATURES OF THE SDK

The functionalities and features can be categorized in four main groups.

Advanced addressing and routing. The core concept of the SDK is geographical addressing and routing. The SDK implements protocols for wireless multi-hop communication

²This requires to operate network interfaces in promiscuous mode.

³The originator of a packet is referred to as source, and the last forwarder as sender.

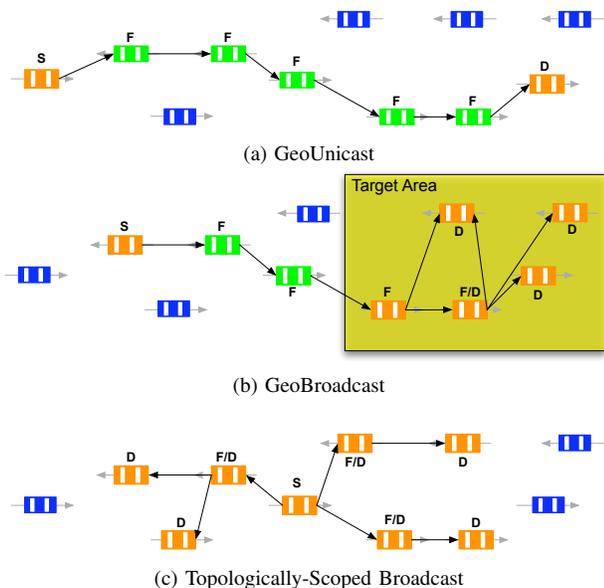


Fig. 2. Forwarding Types in Geo-Networking

utilizing geographical positions (Geo-networking). It is further enhanced by various functionalities:

- Periodic timer-controlled transmission of network-layer beacons,
- Search of a node's location by means of a multi-hop query/reply-based location service,
- Efficient flooding of data packets by means of link-layer broadcast and suppression of duplicated packets based on a packet's source address and sequence number,
- Prioritization of packet transmission based on application requirements by means of flexible queuing and scheduling of packets,
- Temporary caching of data packets in sparsely connected networking scenarios (store-and-forward),
- Distributed control of bandwidth consumption for periodic network-layer beacons by means of transmit power control (TPC) with fair assignment of bandwidth among nodes,
- Flexible cross-layer information exchange via the *information connector*.

Geographical routing and IP integration. In addition to directly delivering safety messages, the SDK also supports IP-based applications. The implemented functions cover:

- An unified addressing scheme for MAC, Geo-networking and IP version 4/6 addresses which enables scalable address autoconfiguration,
- Support of standard IP-based applications for both IPv4 and IPv6 by encapsulating IP packets into Geo-networking packets and tunneling over the ad hoc network.

Security and anonymity. The SDK features advanced algorithms to provide security and privacy along with innovative solutions to check the trustworthiness of communication peers. The following functions have been implemented:

- Cryptographic protection of data transmission from malicious alteration by means of digital signatures and certificates for authentication, integrity and non-repudiation,
- Anonymity by means of temporary node identifiers (pseudonyms) and control of pseudonym usage and their change by an application-layer instance,
- Plausibility checks to compare received packet header data from other nodes (timestamp, distance, speed) with expected values and boundaries by heuristics,
- Estimation of the trustworthiness of nodes based on authentication and plausibility checks; isolation of untrustworthy nodes from network operations.

Enhanced radio control. In order to support advanced communication features, the SDK has implemented special functions⁴ to interact with wireless drivers for sending and receiving, including:

- Control of radio parameters on a per-packet basis, such as transmit power and channel number,
- Utilization of link-layer information for routing, i.e., link state, signal strength and failed link-layer transmissions.

IV. IMPLEMENTATION, CONFIGURATION AND USAGE

The *CAR-2-X Communication SDK* is developed for 32-bit x86 CPU architectures, but has been ported to other hardware platforms including MIPS. The minimum recommended CPU features, for usage of the *CAR-2-X Communication SDK* without data security, are a 200 MHz processor and a math co-processor. While executed, the SDK consumes about 1 MB RAM Resident Memory, whereas the recommended minimum RAM size is 64 MB. The SDK is developed for the *Linux* operating system, running completely in user space. The SDK supports *Redhat, Fedora, SuSe, Ubuntu, Debian* distributions. The currently recommended *Linux* kernel version is 2.6.19 and relies on publicly available software libraries.

The software is comprised of two main modules: The *CAR-2-X protocol stack* is implemented as a user-space daemon and realizes the various features listed in Sec. III. The *CAR-2-X Application Programming Interface (API)* provides access to the protocol stack. The API implementation comprises message format definitions and functions to parse and assemble messages, abstract functions for sending and receiving messages, and access to management and configuration interfaces. The API allows applications to be executed remotely on dedicated application units (such as PDAs), which can run under various operating systems. The SDK also contains a number of additional example applications and various helper scripts and facilities. The software is provided as a single-file *Linux* package for different package managers (*RPM* and *DEB*).

The *CAR-2-X* protocol stack offers different interfaces to the applications for both data and management (Fig. 3):

Data. Applications specify addressing type, target, payload. The interface is UDP-based, so applications can run on a

⁴Requires specific support from IEEE 802.11 hardware and driver.

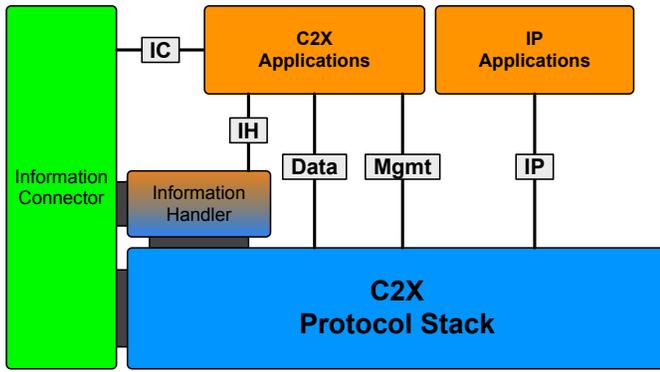


Fig. 3. Interfaces of the CAR-2-X Protocol Stack

separated host. On the receiver's side, applications receive payload and header fields.

Management (Mgmt). The SDK allows to get and set the current geographical position, to set pseudonyms and security credentials, to get location table entries and to enforce a location service for a specific node.

Information Connector (IC). The IC enables structured and efficient information exchange across layers. It can be used for asynchronous exchange of event based on a publish/subscribe scheme, where applications can subscribe to certain services and are notified when events occur (for example, when a new neighbor node appears).

Information Handler (IH). The IH aggregates dynamic information from multiple applications and appends these information to the payload of periodic messages,

IP. The SDK supports legacy/popular applications based on IPv4/v6, including addressing and routing. Provides automatic address configuration for application units.

The *CAR-2-X Communication SDK* can be executed in two main configurations: In the *co-located configuration*, applications and the CAR-2-X protocol stack are executed on the same hardware. In the *remote configuration*, applications and the protocol stack run in different systems that are interconnected, typically by Ethernet. The latter configuration facilitates the usage of alternative operating systems than *Linux*.

The protocol stack can be flexibly configured by means of configuration files that allow to enable and disable various features of the system, including wireless parameters (channel, channel width, data rate, per-packet control), node-specific parameters (node addresses, on-board vs. road-side unit), and functional parameters (beacon interval, security, plausibility checks, trustworthiness assessment, congestion control mechanisms and others). The protocol stack provides capabilities for event logging and packet tracing. Events are logged in files that can be used for off-line evaluation. Type and level of debug information can be adapted to user needs.

The protocol stack can be configured to use different wireless, wired and even virtual network interfaces. The latter case allows to create a test network of virtual machines execute instanced of the SDK. With commercial wireless interfaces, such as IEEE 802.11a, the SDK can run in standard personal

computers as an environments for application development. Finally, the SDK can be executed in embedded hardware providing access to in-vehicle interfaces (such as CAN) and featuring IEEE 802.11p radio interfaces as available in first prototype hardware platforms.

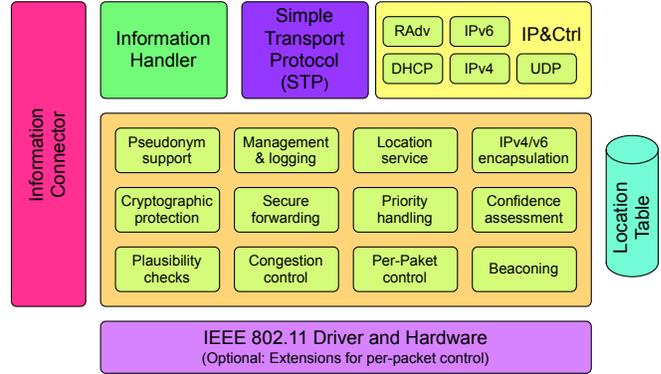


Fig. 4. Implementation Design

V. FRAMEWORK FOR PERFORMANCE EVALUATION

The *CAR-2-X Communication SDK* is a software environment that allows for experiments related to vehicular communication. It can be applied to study the performance of communication protocols and its effectiveness for vehicular applications, in particular for road safety applications. Potential studies can be conducted in different environments, such as cities, rural roads and highways, in order to address the specific influence of the radio propagation conditions on the performance. The SDK also allows for evaluation of the effect of advanced communication mechanisms and algorithms at different layers of the protocol stack; examples are resource allocation, data congestion control, security and others. Moreover, the possibility to execute the SDK on hardware platforms can give valuable results for dimensioning of hardware components of future on-board or communication units.

Consequently, we identify basically three different directions of experiments with the SDK, namely the performance evaluation of network and transport layer protocols that are implemented by the SDK, the study of cross-layer optimizations between radio and networking layers and between network and application layers; and finally the benchmarking of hardware platforms. Tab. I gives a list of selected configuration parameters that can be varied in the SDK for experimentation.

Performance evaluation of network and transport layer protocols. Typical studies are related to the broadcasting of heartbeats to neighbors and the multi-hop packet transport with GeoBroadcast or GeoUnicast. Outcomes of experiments include classical performance metrics such as throughput, packet loss rate, delay and jitter, but also specific metrics such as packet reception probability for heartbeats and reliability for multi-hop packet transport (GeoBroadcast, GeoUnicast). The observation of these metrics implicitly allows to assess the communication overhead for packet processing, signalling

Parameter	Value
Node type	OBU, RSU
Type of radio interface	IEEE 802.11a,b,p; generic
Channel frequency	2,4GHz, 5,4GHz, 5,9GHz
Channel bandwidth	10, 20MHz
Data rate	3–54Mbps
Heartbeat send interval	10–...ms
Promiscuous mode	On/off
Per-packet radio control	On/off
Cryptographic protection	On/off
Plausibility checks	On/off
TPC for heartbeats, for GeoUnicast	On/off, on/off
Packet caching (Store&forward)	On/off
Min/max transmit power, steps	Radio specific [dBm]
Distance-based packet suppression	On/off
Suppression range	0–...m
Tracing; debug level	On/off; 1–5
Prioritization	On/off/configurable

TABLE I
SELECTED CONFIGURATION PARAMETERS

and packet headers. The performance evaluation of IP-related protocols (IPv4, IPv6, UDP, TCP, IPSec, TLS) on top of Geo-networking and their overhead assessment is important for infotainment applications.

In addition to general performance studies, selected mechanisms and algorithms can be evaluated and their impact on the performance studied. Of particular importance are mechanisms for data congestion control, such as transmit power control, where the SDK allows to evaluate of state-of-the-art algorithms, such as DFPAV and power-aware greedy forwarding [15], [16]. Other examples are the study of advanced algorithms such as cryptographic protection of Geo-networking packets by digital signatures and certificates, prioritization of packets with enhanced queuing&scheduling, and temporary caching of packets (store&forward) for packet forwarding in sparse network scenarios.

Cross-layer optimizations. The SDK allows to study performance effects of cross-layer optimizations. First, *application-layer metrics* can be evaluated in order to understand the effect of network and transport layer performance on the application-layer metrics, such as message delay of a *cooperative forward collision warning* or an *emergency vehicle warning*. Such metrics may also include user-specific metrics as perceived by a driver via the Human-Machine Interface (HMI). Another group of cross-layer optimizations are related to network/transport and radio layers: With the SDK, experimental systems with different types of antennae, RF cables and connectors can be tested and its performance evaluated. Moreover, various parameters of the radio interface can be varied (data rate, frequency, channel bandwidth, inter-frame spaces, RTS/CTS, etc.) and its impact on upper layer metrics assessed. Finally, cross-layer congestion control techniques can be examined.

Benchmarking of hardware platforms. The SDK can be executed as a reference software implementation for vehicular

communication in order to assess the relative performance of different hardware platforms, such as Car PCs and various automotive platforms and by running a set of standard performance tests as described above. The result of the benchmark tests can determine the minimal demands on the system and on individual components (such as processor and co-processors, memory and interfaces) in order to meet the low-cost requirements of the automotive industry.

VI. CONCLUSIONS

The *CAR-2-X Communication SDK* is a software implementation of the protocol stack for ad hoc networking using IEEE 802.11 radio. Core networking concept is the use of Geo-networking. This core solution is complemented by various functions to meet requirements from safety and infotainment applications, including advanced addressing and routing, integration of Geo-networking and Internet protocols IPv4/6, security and anonymity, and enhanced radio control. The SDK enables experiments with CAR-2-X communication protocols in different environments and facilitates rapid development of safety and infotainment applications.

REFERENCES

- [1] NEC CAR-2-X Communication SDK. <http://c2x-sdk.neclab.eu>, last accessed 19. February 2009.
- [2] A. Festag et al. NoW - Network on Wheels: Project Objectives, Technology and Achievements. In *Proc. WIT*, Hamburg, Germany, March 2008.
- [3] A. Hiller, A. Hinsberger, M. Strassberger, and D. Verburg. Results from the WILLWARN Project. In *European ITS Congress*, June 2007.
- [4] A. Festag, H. Füssler, H. Hartenstein, A. Sarma, and R. Schmitz. FleetNet: Bringing Car-to-Car Communication into the Real World. In *11th ITS World Congress and Exhibition*, October 2004.
- [5] CVIS. Cooperative Vehicle-Infrastructure Systems. <http://www.cvisproject.org>, last accessed 19. February 2009.
- [6] SAFESPOT. Cooperative Vehicles and Road Infrastructure for Road Safety. <http://www.safespot-eu.org>, last accessed 19. February 2009.
- [7] F. Kargl, P. Papadimitratos, and L. Buttyan. Secure Vehicular Communication Systems: Implementation, Performance, and Research Challenges. *IEEE Communication Magazine*, 46(11):110–118, 2008.
- [8] GEONET. <http://www.geonet-project.eu>, last accessed 19. February 2009.
- [9] COMeSafety. European ITS Communication Architecture – Overall Framework – Proof of Concept Implementation. Version 2.0, October 2008. Available at <http://www.comesafety.org>, last accessed 19. February 2009.
- [10] ETSI Technical Committee ITS. <http://www.etsi.org>, last accessed 19. February 2009.
- [11] PRE-DRIVE C2X. PREparation for DRIVING implementation and Evaluation of C-2-X communication technology. <http://www.pre-drive-c2x.eu>, last accessed 19. February 2009.
- [12] CAR-to-CAR Communication Consortium. C2C-CC Manifesto. Version 1.1, August 2007. Available at <http://www.car-to-car.org>, last accessed 19. February 2009.
- [13] CAR-to-CAR Communication Consortium. Forum and Demonstration 2008, October 2008. <http://www.car-to-car.org/index.php?id=137>, last accessed 19. February 2009.
- [14] B.N. Karp and H.T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. MobiCom*, pages 243–254, Boston, MA, USA, August 2000.
- [15] M. Torrent-Moreno. *Inter-Vehicle Communication: Achieving Safety in a Distributed Wireless Environment, Challenges, Systems and Protocols*. PhD thesis, University Karlsruhe, Germany, July 2007. <http://www.uvka.de/univerlag/volltexte/2007/263/>, last accessed 19. February 2009.
- [16] A. Festag, R. Baldessari, and H. Wang. On Power-Aware Greedy Forwarding in Highway Scenarios. In *Proc. WIT*, pages 31–36, Hamburg, Germany, March 2007.