Vespa: Emulating Infotainment Applications in Vehicular Networks

The Vehicular Emulations Platform for Real Applications (Vespa) was developed to allow for experiments with many vehicles running real-world applications with real-time requirements in complex network scenarios. A use case on the performance of video-streaming applications demonstrates Vespa's usefulness.

Sergi Reñé, Juanjo Alins, Jorge Mata-Díaz, Carlos Gañan, Jose L. Muñoz, and Oscar Esparza
Technical University of Catalonia, Barcelona

Applications for vehicular networks are grouped into safety and nonsafety applications. Infotainment applications—a neologism for *information and entertainment*—are included in the latter group and include Internet access, multiplayer games, multimedia applications, chat functions, and videoconferencing. Safety and nonsafety applications have different requirements. Most safety applications, for example, are based on data dissemination in a nearby geographical area, while most nonsafety applications require a network-layer mobility solution to give vehicles Internet access.

As vehicles in motion change their attachment point, Mobile IP is expected to provide session continuity and global reachability. Such mobility protocols provide seamless connectivity to a peer moving between different subnets or domains. Although researchers have proposed mobility management protocols to provide continuous Internet connection, the high mobility of vehicles creates frequent handoffs, which can result in significant packet delay and packet loss. Researchers and developers thus need a framework to evaluate protocols and services in this challenging vehicular scenario.

Obviously, the most reliable framework would include outdoor experiments to evaluate how applications behave under real conditions. However, such a framework is extremely costly and has several drawbacks due to the difficulty of managing a car fleet. Software platforms can therefore play a vital role in testing real-world scenarios, and most research in vehicular networks relies on simulations. The problem, however, is that these simulation platforms focus mainly on providing a testing framework for safety applications that don't require infrastructure, so it's difficult to assess infotainment applications. Also, integrating real-world implementations of applications within these simulators is challenging, and few conventional simulators can emulate networks in real time. (For more information, see the "Related Work in Testing Vehicular Networks" sidebar.)

Simulators use discrete events to efficiently execute network events in batch. In contrast, emulators use a scheduler that ties event

Published by the IEEE CS ■ 1536-1268/14/$31.00 © 2014 IEEE

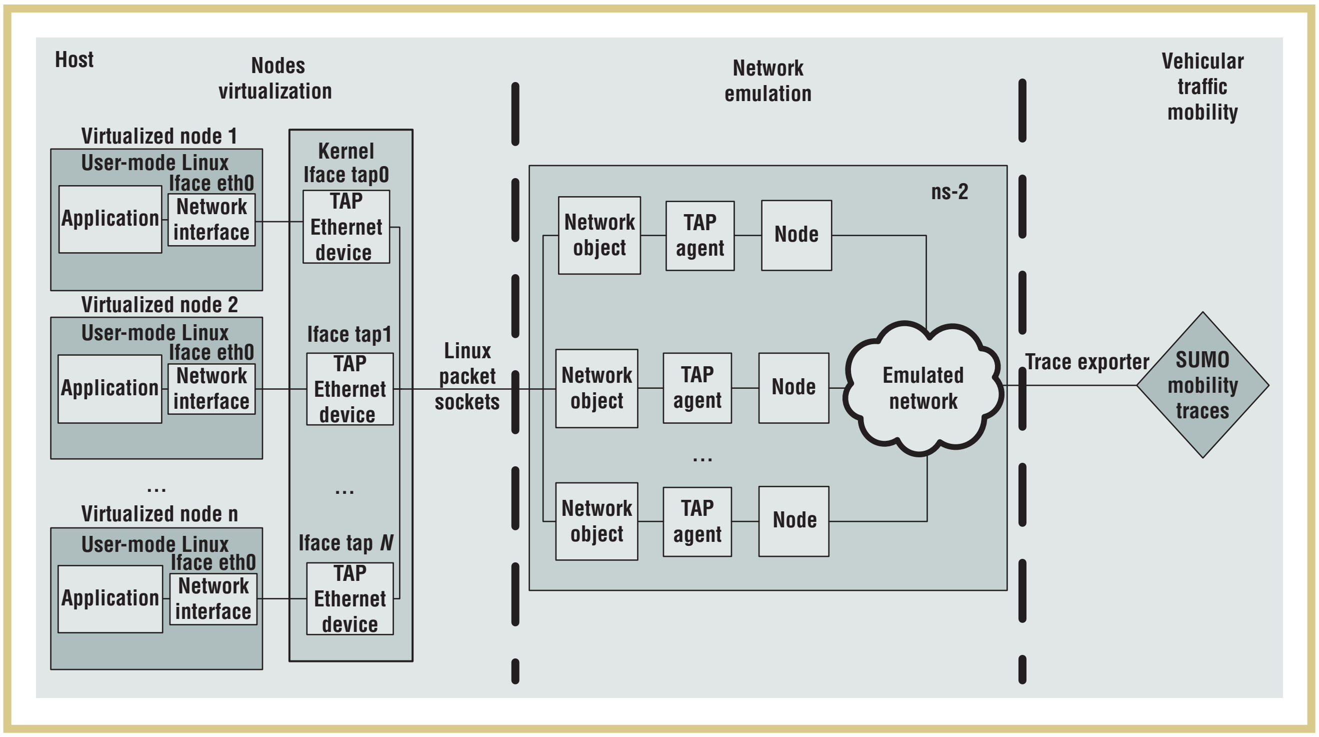


Figure 1. Vehicular Emulations Platform for Real Applications (Vespa). To connect virtualized nodes with ns-2, Vespa uses TAP virtual Ethernet devices (http://vtun.sourceforge.net) to transport information to and from UML virtual machines.

execution with real time. This makes emulators less scalable compared to simulators, but in our scenario, emulation is essential for testing real-world implementations of infotainment applications, especially those with real-time requirements. Here, we present our *V*ehicular *E*mulations *P*latform for Real *A*pplications (Vespa), a real-time emulator for testing infotainment vehicular applications with infrastructure participation.

Vespa

Vespa focuses on providing testing capabilities for real-time Internet-based services or centralized services in vehicular networks that require infrastructure. Vespa consists of a set of software developments and a GUI tool that integrates three basic modules: a node virtualization module based on User Mode Linux (UML) virtual machines (see http://user-mode-linux.sourceforge.net), the network simulator 2 (ns-2) using its emulation features, and the Simulation of Urban Mobility (SUMO) road traffic mobility simulator.

Vespa's main novelty in relation to other vehicular networks evaluation tools is that it captures key aspects needed to test real-world implementations of real-time applications in vehicles:

- Virtualized nodes let us install and test real-world software in the virtual machines, which represent the onboard unit (OBU) at vehicular nodes.
- ns-2's emulation features let us introduce these vehicular nodes into a live network.
- The SUMO simulator lets us feed the ns-2 emulator with realistic information about nodes' mobility.

Vespa also makes it possible to emulate both roadside units (RSUs) and server nodes in the infrastructure domain, which are essential for the proper working of most infotainment services.

To the best of our knowledge, Vespa is the only emulator that offers the possibility of testing applications' network mobility in real-time, including IP micro-mobility protocols, such as Fast Handoffs for Mobile IP (FMIP). IP micro-mobility protocols complement Mobile IP by offering fast and seamless handoff control in limited geographical areas, and IP paging to support scalability and power conservation. These protocols add an interesting feature to our emulation platform, letting it test applications that require seamless handoffs for successful deployment in vehicular environments with an infrastructure; in such applications, disruptions caused by network-layer handoffs can seriously impair the users' experience.

As Figure 1 shows, Vespa comprises three main logic modules.

Nodes Virtualization

This module consists of nodes virtualized with UML machines, which

Related Work in Testing Vehicular Networks

As Table A shows, several tools have been proposed to evaluate vehicular network performance.

Michal Piórkowski and colleagues proposed the Traffic and Network Simulation Environment (TraNS),[1] which, like the *Ve*hicular Emulation*s* *P*latform for Real *A*pplications (Vespa), integrates the Simulation of Urban Mobility (SUMO) road traffic mobility simulator and the ns-2 network simulator. TraNS includes a Traffic Control Interface (TraCI) module that allows feedback between the mobility model and the vehicle behavior. This feature is extensively used for safety and traffic efficiency applications. For example, when a vehicle broadcasts information reporting an accident, neighboring vehicles might slow down. However, this isn't the case for infotainment applications, such as Internet access, multiplayer games, and multimedia applications, so Vespa doesn't implement this feature. Also, TraNS is a simulator, not an emulator; it can't virtualize nodes for running implemented applications, and it doesn't support IP micro-mobility protocols like Vespa does.

Christoph Sommer and colleagues developed the Vehicles in Network Simulation (Veins) framework (http://veins.car2x.org),[2] which pairs SUMO with OMNet++ through a TCP connection as TraCI does. Veins provides interesting features for safety

TABLE A
Comparing vehicular network simulators and emulators.

Features	TraNS*	Veins†	NCTUns‡	Twine	iTetris	Vespa
Platform	ns-2	OMNet++	NCTUns	Twine	ns-3	ns-2
Traffic simulator	SUMO	SUMO	NCTUns	None	SUMO	SUMO
Emulation	No	No	Yes	Yes	No	Yes
IP mobility	No	No	Yes	No	No	Yes
IP micro-mobility	No	No	No	No	No	Yes
802.11p	Yes	Yes	Yes	No	Yes	Yes
Infrastructure capabilities	No	No	No	Yes	Yes	Yes
Orientation	Safety apps	Safety apps	Both	Infotainment	Both	Infotainment
Traffic control interfacet (TraCI)	Yes	Yes	No	No	Yes	No

**TraNS* = Traffic and Network Simulation Environment
†*Veins* = Vehicles in Network Simulation
‡*NCTUns* = EstiNet Network Simulator and Emulator

host the applications to be tested using Vespa. Node virtualization is expensive in terms of resources (including CPU, memory, and storage), so the number of guest machines that can be virtualized in a particular host is limited. However, rather than virtualize all the nodes in the vehicular network, Vespa lets us virtualize only those in which we want to test real-world software. The rest of the nodes can be emulated within the ns-2.

Network Emulation

Virtualized nodes are connected through an ns-2 emulated vehicular network. To connect virtualized nodes with ns-2, Vespa uses TAP virtual Ethernet devices (http://vtun.sourceforge.net) to transport information to and from UML virtual machines. As Figure 1 shows, a TAP device is assigned to each UML, which connects the virtual node to the emulated network. The UML machines consider TAP devices to be common Ethernet devices that are directly connected to the corresponding virtual Ethernet interface.

To emulate the network, Vespa uses ns-2's emulation feature of ns-2. ns-2 uses a soft real-time scheduler that ties event execution within the simulator to real time and network objects, and it uses TAP agents to connect the emulated network with a live network. We use ns-2 emulation extensions,[1] which are part of ns-2's code, to implement the network objects and TAP agents in Vespa, and we use the network objects to send and receive packets to and from a live network. The network objects read packets from and write packets to TAP network devices at the link layer using Linux packet sockets. The TAP agents are application-level processes on ns-2 nodes that convert network packets between the emulated wireless network and the real network using a network object to access a network device on the link layer (TAP virtual interfaces). Each TAP agent can be connected to at most one applications, but the lack of support for network infrastructure and network emulation features prevents testing real-world and infotainment applications.

The EstiNet Network Simulator and Emulator (NCTUns) is a discrete-event network simulator based on ns-2 that provides a complete GUI tool to configure test scenarios easily.[3] Using its car agent, NCTUns can be used as a vehicular network simulator. It also supports emulation features. Despite NCTUns' advantages—including ease of use—it has several drawbacks. One drawback is that it supports only a fixed, predetermined number of vehicles per simulation. Although NCTUns provides some random speed models, it doesn't use SUMO, which provides more realistic models. Moreover the simplification of some of its models can produce inaccurate results compared with other simulators. NCTUns' main drawback is that it requires installation of a Fedora 9 Linux distribution, which limits its usage. Finally, NCTUns is not as scalable as other platforms (see Figure 2c in the main text).

Another vehicular approach is the iTetris platform,[4] which consists of SUMO, an ns-3 network simulator, and an application module all connected by the iTetris Control System (iCS) module. Applications can be independently implemented and run on the top of iCS using the applications block. Triggered by application commands, ns-3 simulates vehicular transmissions. Applications are notified of receptions deriving from these communications; they then produce actions to be undertaken in the road traffic scenario simulated by SUMO using TraCI. iTetris lets developers synchronize simulation time with the application, traffic, or wireless communications events. However, iTetris can't support running a real-world application because it uses its own API to create network sockets. Thus, unlike with Vespa, users cannot test real-world applications using an actual operating system. Moreover, ns-3 doesn't support IP mobility, micro-mobility, Mobile IP, or Fast Handoffs for Mobile IP (FMIP) protocols. For this reason, iTetris is unsuitable for analyzing infotainment applications in which handoffs and mobility are primary issues.

All of these approaches are vehicular simulators. The literature reports on other tools that can run real-world applications on modeled networks. In this respect, the Twine[5] emulator is one of the most popular. Twine targets realistic, scalable, and flexible evaluation of wireless technologies and applications. Twine is oriented to test wireless networks (wireless local area networks, mesh networks, or mobile ad hoc networks) but isn't oriented to test vehicular networks. Thus, Twine can't use realistic mobility models needed to evaluate vehicular applications.

REFERENCES

1. M. Piórkowski et al., "TraNS: Realistic Joint Traffic and Network Simulator for VANETs," *ACM Mobile Computing and Communications Rev.*, vol. 12, no. 1, 2008, pp. 31–33.

2. C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Trans. Mobile Computing*, vol. 10, no. 1, 2011, pp. 3–15.

3. S. Wang and C. Chou, "NCTUns Tool for Wireless Vehicular Communication Network Researches," *Simulation Modelling Practice and Theory*, vol. 17, no. 7, 2009, pp. 1211–1226.

4. V. Kumar et al., "iTetris: Adaptation of Its Technologies for Large-Scale Integrated Simulation," *Proc. 2010 IEEE Vehicular Technology Conf.* (VTC 10), 2010, pp. 1–5.

5. J. Zhou, Z. Ji, and R. Bagrodia, "Twine: A Hybrid Emulation Testbed for Wireless Networks and Applications," *Proc. IEEE Ann. Joint Conf. IEEE Computer and Comm. Societies* (INFOCOM 06), 2006, pp. 23–29.

network object; TAP agents also implement the address mapping between UML virtual machines, medium-access control (MAC) addresses, and the ns-2 nodes addressing.

Several initiatives are attempting to standardize vehicular communications, including the Car-to-Car Communication Consortium (C2C-CC), IEEE 1609/Wireless Access in Vehicular Environments (WAVE), the European Telecommunications Standards Institute's Intelligent Transport Systems-G5, and Continuous Air Interface for Long and Medium Interface (CALM). All such initiatives use the WAVE physical (PHY) and MAC layers based on IEEE 802.11p, which is an amendment to IEEE 802.11 for the rapidly varying vehicular environment. As a minimum inexpensive solution, a vehicle equipped with only IEEE 802.11 technology can use vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication. Safety and nonsafety applications must be integrated into a single system for a vehicular communication system's successful market introduction. This is why Vespa uses IEEE 802.11 access technology. Various projects improve MAC/PHY implementation in network simulators.[2,3] Vespa uses the 802.11 MAC and PHY model for ns-2 developed by Mercedes-Benz Research & Development North America and the University of Karlsruhe.[2] This model lets us configure many MAC and PHY layer parameters that you can't configure in the ns-2 MAC/PHY implementation; this gives us a higher level of simulation accuracy. Using these extensions lets us use IEEE 802.11p access technology parameters in ns-2. Our default coordination function is the distributed coordination function (DCF).

Vehicular Traffic Mobility

SUMO is the most widely used traffic simulator licensed under GPL. Moreover, it offers several features to build mobility patterns in vehicular networks, including the possibility of creating maps using theoretical models (such as Manhattan grid) or importing real maps from external sources. SUMO also lets us configure aspects related to nodes movement, such as the number of vehicles, maximum speed, and so on. This lets us use Vespa for analyzing the deployability of vehicular services over a real traffic scenario.

SUMO generates netstate dumps that contain information about the nodes' position and speed. These dumps are generated once a mobility scenario is loaded, and they must be converted to suitable mobility traces. A parser module named traceExporter converts netstate dumps to ns-2 mobility traces; ns-2 can then use them as an input to calculate the nodes' network conditions. Traces are calculated once a user introduces a SUMO configuration file to Vespa, or after a user has built a SUMO scenario using our emulation platform's GUI for editing traffic scenarios.

Graphical User Interface

We developed a simple and intuitive GUI to facilitate Vespa use. This GUI makes testing trouble-free and efficient—in contrast to program-driven systems, which require complex programming or scripting. Vespa's graphical software lets you zoom in on low-level parameters without having to go through manuals or specifications. The GUI's clean interface makes it easy to "dive deep" and control the fine details of emulating a complete network. The GUI also helps users edit mobility scenarios and run tests, as it provides a set of utilities that automatically create the configuration files needed for the emulation.

The GUI lets users

- create random maps (including grid maps, spider networks, and totally random maps) and vehicular random routes;
- create maps manually, using an editor that lets them create map nodes and the edges between nodes; and
- import external maps and vehicular routes.

Vespa's graphical software lets you zoom in on low-level parameters without having to go through manuals or specifications.

When users create a manual map, they must also manually edit the routes using another utility that lets them create different routes and assign them to different vehicles.

Once the map and the route configuration files are created, Vespa lets users create a SUMO configuration file. They can use this file to load the mobility scenario a posteriori in the emulation platform. Regarding other parameters, users can configure the RSUs and the network emulation duration; the virtual nodes must be configured, selecting the nodes that must be virtualized using UML virtual machines. ns-2 will directly emulate the other existing nodes. With the RSUs, it's possible to configure the position coordinates and access technology parameters (IEEE 802.11a, 802.11b or 802.11p) of each one using parameters provided elsewhere.[2] All RSUs deployed in the map will be connected to a central server that will be virtualized by a UML virtual machine.

Vespa Accuracy and Scalability

We ran a set of tests to gauge our emulation platform's performance limitations. In particular, we assessed the impact on the emulation's accuracy, because it inevitably introduces delay when network traffic is sent between applications running on the virtual machines and the real-time simulator. All evaluation tests were performed on a Quad-core (1.6 GHZ) Intel x64 system running Debian-Linux with kernel version 3.2.0-35. The host machine has 32 Gbytes memory, the UML machines use at most 32 Mbytes of RAM, and the simulator's logging process uses 100 Mbytes for the compressed trace file. Inside the UML machines, we used the same Linux kernel as on the host system.

To evaluate Vespa's accuracy, we compared the measurements obtained via emulation with the results of a pure ns-2 simulation. To determine the round-trip times (RTTs), we used simple ping—Internet Control Message Protocol (ICMP) echo—measurements. We evaluated the delays introduced by both the emulated network, including virtualization and the traffic redirection, and the simulation model. We executed the ping command to send 10,000 ICMP packets from one virtualized node to another. Figure 2a shows the results as error bars under various payload sizes. (We increased the kernel's default interrupt frequency such that 1 jiffy becomes 1 millisecond in the modified kernel—that is, so that the accuracy of the packet delay experiment's results is within 1 ms.)

As Figure 2a clearly shows, RTT increases for larger packets, which is explained by the fact that it takes longer to fully place a large packet on the medium. The results also show that the emulation results correspond accurately with the simulation results, with exception of a latency overhead of about 0.15 ms in the emulation case due to the additional packet handling layer in the virtual machine. Note that the standard deviation of Vespa's RTT values is comparable to that of the simulation results.

To evaluate Vespa's scalability, we measured resource requirements in terms of CPU utilization and memory requirements as the number of emulated nodes is increased. The load is generated using the following scenario: vehicles are moving at constant speed, separated by five meters, and communicating with 802.11p. Each node opens a socket and sends fixed-sized *User Datagram Protocol* (UDP) packets to the others at a constant rate. To characterize the overhead, we vary the traffic rate from 50 to 10,000 packets per second (pkt/s) using packet sizes of 100 and 1,000 bytes. We use the `vzmemcheck` command to obtain the memory consumption for the UMLs. For CPU load, we used the `vmstat` command.

As Figure 2b shows, the memory consumption increases linearly with the number of virtual nodes. This is because each virtual machine is a separate executing entity, with constant memory occupancy. On the other hand, the CPU utilization reaches 90 percent with around 20 virtual nodes due to the decrease in the instructions per communicated byte when the number of emulated entities increases. Figure 2c shows the ratio of late packets in terms of packet size. To that end, we virtualized four nodes and used the same traffic pattern as in the other experiments. Compared to Twine, Vespa maintains a lower ratio of late packets; it can therefore better maintain the real-time accuracy during tests than Twine.

Case Study: Video Streaming

To demonstrate Vespa's usefulness, we analyzed a video-streaming service in a vehicular network based on the video playback smoothness and experience. We evaluated only the effects of packet losses caused by network-layer handoffs due to node mobility in a highway scenario; no other aspects—such as transition errors or other sources of information corruption—were considered.

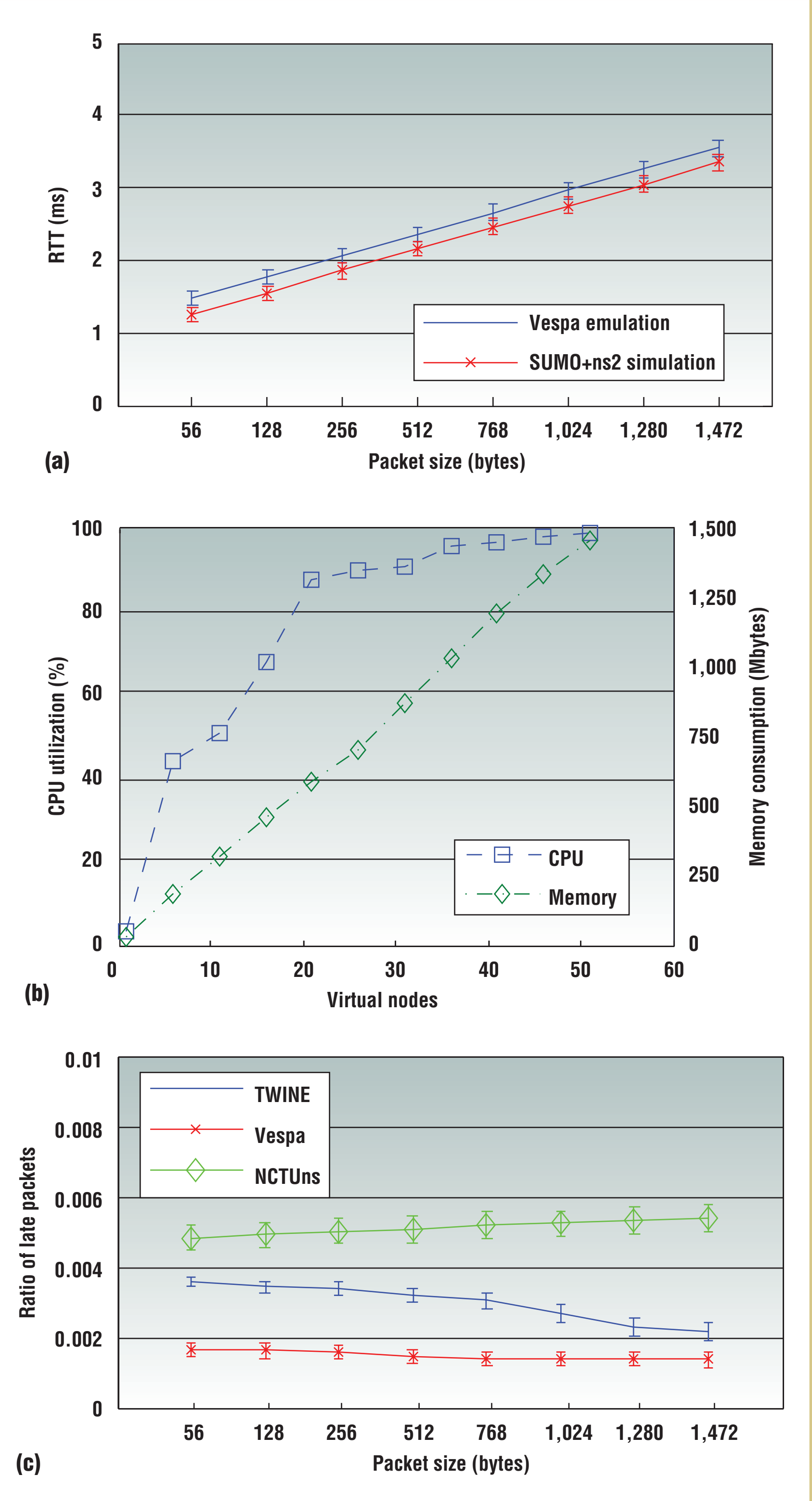


Figure 2. VESPA accuracy and scalability: (a) round-trip times (RTT) with various payload sizes, (b) CPU and memory utilization, and (c) the ratio of late packets. (See the sidebar for more information about Twine and NCTUns.)

TABLE 1
Test parameters.

Parameter Name	Value
Wired links	Bandwidth: 100 Mbits per second (Mbps) Propagation delay: 5 milliseconds
Propagation model	Nakagami
Transmission power	60 megawatts
Frequency	5.85 gigahertz
Data rate	27 Mbps
Basic rate	3 Mbps
802.11 mode	Basic Service Set (BSS)
Coordinate function	Distributed Coordination Function (DCF)
Slot time	13 microseconds (μs)
Short Interframe Space (SIFS)	32 μs
Contention Window (CW) Min	15
Contention Window (CW) Max	1,023
Bandwidth	10 megahertz
Antenna	Omnnidireccional
Distance between roadside units (RSUs)	350 m
Video characteristics	Common Intermediate Format (CIF) 352 $\times$ 288, MPEG-2

Our tests show interesting results regarding service deployment, even in this simplistic scenario. First, handoffs can be performed at the link or network layer. Also, link-layer handoffs can be used when all the RSUs belong to the same IP subnet and the same administrative domain. Network-layer handoffs are most often found in wired and wireless environments in which users carry their mobile devices across multiple IP subnets. This behavior fits in a vehicular environment, in which a vehicular node is moving through different RSUs placed along a road. Thus, we decided to represent the worst case, in which handoffs are performed at network layers, each of which includes a network-layer mobility protocol.

Reference Scenario

The test scenario we designed is an infrastructure scenario in which a set of RSUs are deployed over a highway in an overlapped manner avoiding coverage blackouts in the road. All the RSUs are connected to a central router, which is connected to a video-streaming server. Both the infrastructure domain's video-streaming server and the vehicular node with the video player are emulated by UML virtual machines. No other vehicles are virtualized with UML or emulated with ns-2.

The Live555[4] libraries provide the testbed's multimedia applications. The video transmitted during the tests has a Common Intermediate Format (CIF) format (352 × 288) encoded at constant bit rate (CBR) with 1,000 Kbytes per second and 25 frames per second. The video-streaming media is sent using the UDP in conjunction with the Real-time Transport Protocol (RTP), so the end-to-end communication is unreliable. Table 1 shows other parameters used during the tests.

Performance Results

The objective of these tests was to assess the impact of (only) network-layer handoffs on the quality of a video clip streamed to a (single) vehicle on a highway. During these handoffs, packet losses will affect the streamed video's quality. We analyze the streamed video's quality using the Peak Signal-to-Noise Ratio (PSNR) between the original video and the video transmitted within the vehicular network and received by the vehicular node.

To calculate the PSNR, which is conclusively valid only when used to compare results from the same encoder and content, we use an MPEG-2 decoder that recovers the lost gaps of lost frames during the communication by representing the previous frame received. Figure 3a shows the PSNR for three different speeds between 20 and 40 meters per second (m/s), which are coherent speeds for a highway scenario. Figures 3b and 3c represent the video disruption time during the playback for Mobile IP and FMIP handoffs; they also show the video degradation during handoffs using both Mobile IP and FMIP.

As Figure 3a shows, during handoffs, the PSNR obtained is lower than 20 decibels (dB). These PSNRs fall during video playback, causing the video player to freeze when the PSNR is that low. Thus, during handoffs, the quality level is insufficient due to the missing frames and choppy playback. When no handoffs occur—as in our particular scenario, which had no other types of losses during video transmission—the PSNR value is infinite (though Figure 3a shows this PSNR value as 100 dB). In a real test that considered other network aspects, such as transmission errors, PSNR values between 30 and 50 dB would be considered a good quality level and video playback would be smooth. According to Figure 3a, increasing the vehicular node's speed not only produces more handoffs in the same time (as its speed increases, the vehicle traverses more RSUs), but these handoffs also increase their duration. So, increasing the speed causes the video player to be frozen more often and for longer periods.

Figure 3 also offers some conclusions about Mobile IP (Figure 3b) and FMIP use (Figure 3c). FMIP is supposed to reduce the handoff delay by either introducing L2 triggers to anticipate the handoff or by managing most of the handoff operations inside a local domain. Minimizing the handoff delay, the FMIP standard pretends to reduce the amount of lost packets during the handoff. As Figure 3c shows, FMIP isn't very fast, and it doesn't always guarantee a successful fast handoff in most cases. In fact, at high speeds, the improvement between using FMIP and MIP is insignificant. This is because there are more unsuccessful FMIP handoffs as vehicle speed increases. The increment of unsuccessful FMIP handoffs is caused by any message loss in the FMIP handoff-anticipation mechanisms. These failures occur when a handoff between subnets is achieved before new network parameters are configured and before the tunnel is established to forward the packets on the fly. These failures cause packet loss and happen more often with the increase of node velocity.

According to our tests, it seems impossible to offer a video-streaming service on a highway with sufficient quality when a UDP/RTP transmission is used. The video playback degradation is unacceptable in most of the cases because MIP and FMIP handoffs are too slow. Also, here we've analyzed only the effects of L3 handoffs; adding other important aspects—such as transmission errors, losses due to congestion, and bandwidth variations due to weather effects—would only worsen the system's performance.

The research required to alleviate these problems and provide an acceptable service is beyond our article's scope. However, we can foresee possibilities to reduce losses during handoffs. For example, using reliable transport protocols such as TCP would recover gaps due to lost frames at the expense of adding delays. Also, another possibility might be to use advanced video-coding techniques, such as application-layer forward error

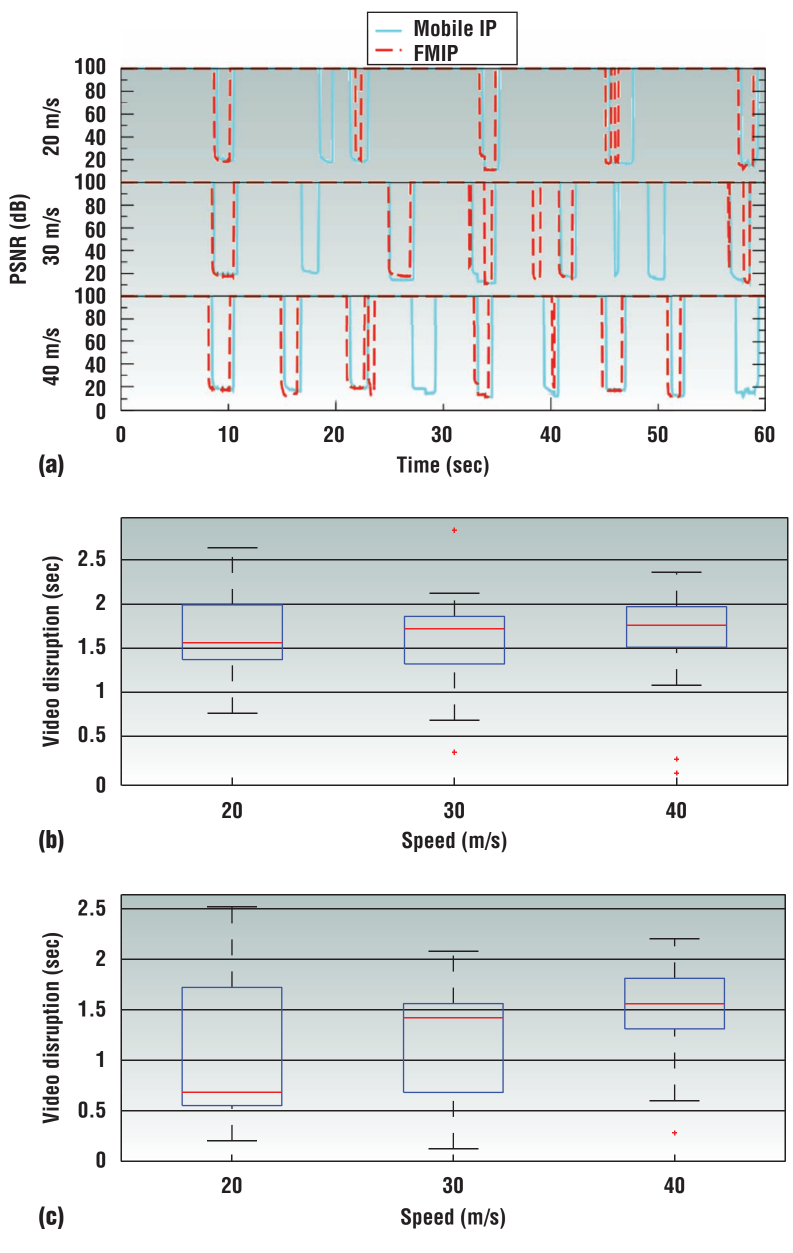


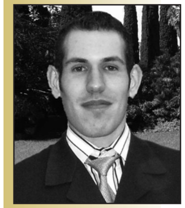
Figure 3. Case study performance results. A constant bit rate (CBR) 1,000 Kbps video transmission uses the User Datagram Protocol (UDP) in conjunction with the Real-time Transport Protocol (RTP) for different vehicle speeds: (a) Peak Signal-to-Noise Ratio, (b) video disruption time using Mobile IP, and (c) video disruption using as Fast Handoffs for Mobile IP (FMIP).

the AUTHORS

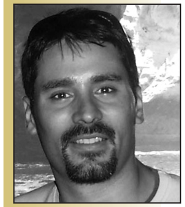
Sergi Reñé is a doctoral student and member of the Department of Telematic Engineering's Telematics Services Research Group at the Technical University of Catalonia, Barcelona. His research interests include cross-layer transport protocols, multipath transport protocols, and vehicular communications. Reñé has an MS in telematics from the Technical University of Catalonia. Contact him at serge.rene@entel.upc.edu.

Juanjo Alins is a professor and research staff member in the Department of Telematic Engineering's Telematics Services Research Group at the Technical University of Catalonia, Barcelona. His research interests include network services to the home, audiovisual appliances, streaming QoS, secure multimedia transmission, traffic modeling, and statistical performance analysis. Alins has a PhD in telecommunications engineering from the Polytechnic University of Catalonia, Spain. Contact him at juanjo@entel.upc.edu.

Jorge Mata-Díaz is a research staff member in the Department of Telematic Engineering's Telematics Services Research Group at the Technical University of Catalonia, Barcelona. His research interests include network services to the home, audiovisual appliances, streaming QoS, secure multimedia transmission, traffic modeling, and statistical performance analysis. Mata-Díaz has a PhD in telecommunications engineering from the Polytechnic University of Catalonia, Spain. Contact him at jmata@entel.upc.edu.



Carlos Gañan is a postdoctoral researcher in the Department of Telematic Engineering's Information Security Group at the Technical University of Catalonia, Barcelona. His research interests include multimedia communications, network security, and vehicular ad hoc networks. Gañan has a PhD in telematics from the Technical University of Catalonia. Contact him at carlos.ganan@entel.upc.edu.



Jose L. Muñoz is an associate professor in the Department of Telematic Engineering at the Technical University of Catalonia, Barcelona. His research interests include security and privacy in computer networks. Muñoz has a PhD in network security from the Technical University of Catalonia. Contact him at jose.munoz@entel.upc.edu.

Oscar Esparza is an associate professor in the Department of Telematic Engineering at the Technical University of Catalonia, Barcelona. His research interests include identity, trust and reputation in p2p networks, network security in VANET, and TCP performance over satellite links. Esparza has a PhD in mobile agent security and network security from the Technical University of Catalonia. Contact him at oesparza@entel.upc.edu.

correction (FEC) with rateless erasure coding combined with Scalable Video Coding (SVC) extension of H.264/MPEG4-AVC.[5,6]

Vespa is primarily aimed at researchers and developers who want to easily test applications in a vehicular scenario. Vespa and a set of video testing tools developed for this test-bed are freely available can be easily installed in Linux systems or downloaded as a live CD and executed in any desktop computer (see http://sourceforge.net/projects). Vespa could be used, for example, to compare how various codification techniques (or video players) perform in a controlled vehicular scenario. Vespa makes it possible to test applications using the same software developed for desktop computers without requiring extensive modeling time for network simulators, while also avoiding the limitations caused by simplified application behaviors. P

ACKNOWLEDGMENTS

Our work was supported in part by the Spanish Research Council, project TEC2011-26452 (Servet); the Spanish Ministry of Science and Education, project Consolider CSD2007-00004 (ARES); Generalitat de Catalunya, grant 2009 SGR-1362 to consolidated research groups; the Secretaria d'Universitats i Recerca of the Departament d'Economia i Coneixement; and the European Social Funds.

REFERENCES

1. D. Mahrenholz and S. Ivanov, "Real-Time Network Emulation with ns-2," *Proc. 10th IEEE Int'l Symp. Distributed Simulation and Real-Time Applications Proc. Dynamic Soft Real Time* (DS-RT 04), 2004, pp. 29–36.
2. Q. Chen et al., "Overhaul of IEEE 802.11 Modeling and Simulation in ns-2," *Proc. 10th ACM Symp. Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, 2007, pp. 159–168.
3. J. Mittag et al., "Enabling Accurate Cross-Layer Phy/Mac/Net Simulation Studies of Vehicular Communication Networks," *Proc. IEEE*, vol. 99, no. 7, 2011, pp. 1311–1326.
4. R. Finlayson, "Live555 Streaming Media," Live555, 2009; www.live555.com/liveMedia.
5. T. Schierl et al., "SVC-based Multisource Streaming for Robust Video Transmission in Mobile Ad Hoc Networks," *IEEE Wireless Comm.*, vol. 13, no. 5, 2006, pp. 96–103.
6. S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Adaptive Unicast Video Streaming with Rateless Codes and Feedback," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 20, no. 2, 2010, pp. 275–285.