To appear in the Proceedings of the 6th Workshop on Hot Topics in Operating Systems, Cape Cod, MA, 5-6 May 1997

The Role of Trace Modulation in Building Mobile Computing Systems

M. Satyanarayanan and Brian Noble
School of Computer Science
Carnegie Mellon University

Abstract

In this paper we put forth the view that trace modulation is an indispensable technique for building and evaluating mobile computing systems. The essence of our solution is transparent, real-time, trace-driven emulation of a target network. Although conceptually simple, trace modulation strikes an attractive balance between the conflicting demands of realism, ease-of-use, and reproducibility. It provides three important benefits. First, it allows control of the complexity of the network environment to which mobile software is exposed. Second, it enables mobile clients to be subjected to reproducible yet realistic network performance. Third, it allows easy exploration of a mobile system in the context of hypothetical mobility patterns and network infrastructures. These benefits suggest that trace modulation will play a central role in the development of future mobile computing systems.

1. Position

Trace modulation, a new technique for network emulation, solves many difficult evaluation and debugging problems induced by mobility. By using a *trace* of network characteristics to drive the emulation, trace modulation strikes an attractive balance between the conflicting demands of *realism*, *ease-of-use*, and *reproducibility*. Consequently, we are convinced that this simple yet potent and versatile technique will play a central role in the development of future mobile computing systems.

2. Problem statement

The network quality experienced by a mobile host can vary dramatically and unpredictably over space and time [6, 12]. This imposes fundamental constraints on the logistics of system development, and on the degree of experimental control for evaluation. On the one hand, the difficulty of accurately re-creating a mobile networking environment *in vitro* argues for live experiments. On the other hand, the difficulty of experimental control in live experiments confounds interpretation of results.

Reproducibility of the networking environment is important for three reasons. First, it is essential for sound performance evaluation of a given system. Second, it is necessary for comparative performance evaluation of alternative mobile system designs. Third, it is valuable in debugging mobile systems because it enables re-creation of conditions that trigger bugs.

The adaptive nature of mobile computing systems [5, 7] makes them especially difficult to debug and evaluate. Modest perturbations in the environment may cause significant changes in the behavior of such a system. This complicates debugging as well as interpretation of measurements in the absence of stringent environmental control.

Environmental interference in wireless networks is both spatially and temporally dependent, and is extremely complex to model and understand. Precise duplication of physical motion is very difficult, yet minor perturbations in the path of a mobile client can sometimes result in substantially different network performance.

Finally, even the mundane logistics of live experiments can pose difficulties. In contrast to wired networks, isolation of the wireless spectrum is very difficult to achieve — mobile clients and other devices which are not part of an experiment may nevertheless have access to the wireless spectrum being used. Weather-related annoyances, such as rain or low temperatures, further constrain live outdoor experiments.

3. Solution: trace modulation

The essence of our solution is to provide application-transparent emulation of a target network on a LAN by modifying the network layer of an operating system. The modified layer drops or delays packets in accordance with a concise, time varying list of performance parameters for a simple network model. Trace modulation thus influences the environment in which a system operates rather than generating the workload for the system.

3.1. Using trace modulation

To use trace modulation, mobile software is run on a LAN-attached host with the modified kernel and a replay daemon. The daemon reads a replay trace, and feeds it to the kernel. The resulting network environment is indistinguishable from that recorded in the trace. However, experiments can be carried out in real time, without mobile network hardware or physical motion.

Software above the modulation layer experiences network performance that is much more reproducible than that obtainable in live experiments. It is important to note that trace modulation is fully transparent to this software. No source or binary changes have to be made, all network traffic into and out of the mobile host is accounted for, and all software runs in real time.

3.2. Producing modulation traces

How are such traces produced? There are three possibilities: *synthetic*, *empirical*, and *hybrid*. A synthetic trace provides a simple, idealized environment. An empirical trace accurately reproduces a single path through a real wireless infrastructure. A hybrid trace consists of a runtime composition of many individual traces, which can be synthetic, empirical or both.

A synthetic trace can be produced manually or through simple programs. Examples of synthetic traces include those in which bandwidth or latency is varied according to a step, impulse, ramp, or other simple function. Using such well-defined functions eases the task of interpreting observations of a complex adaptive system.

Generating an empirical trace is more complex. There are two steps in the process: *trace collection* and *trace distillation*. In the first step, an experimenter with an instrumented mobile host physically traverses a path of interest through some established wireless infrastructure. During this traversal, packets from a known workload are generated. The mobile host records observations of packets as well as network device characteristics. At the end of the traversal, the list of observations represents an accurate trace of network behavior. In the second step we transform this collected behavior trace into a trace suitable for modulation. When replayed, that trace accurately reproduces the network performance seen in the initial traversal.

Support for dynamically constructing hybrid traces is provided by a more sophisticated version of the replay daemon. This incorporates a location simulator that takes a description of a *mobility pattern* and a set of individual traces. The simulator can switch between these traces based on the mobility pattern.

A mobility pattern may be constructed off-line or dynamically generated by another component of the system under test. By appropriate choice of mobility pattern, one can explore the behavior of a system along a new path in an existing wireless infrastructure, or on a path through a hypothetical wireless infrastructure.

4. Key design elements

There are two areas in our design that merit further discussion — the implementation of trace modulation, and the two trace formats. Our main goals in building the replay component have been simplicity and ease of experimentation. The trace formats, while primarily designed for wireless networks, are intended to be useful in other settings as well.

4.1. Implementing trace modulation

The trace modulation support in the kernel is placed between the transport and network layers in the protocol stack; this provides a single point which captures all traffic on the modified host. All packets, regardless of destination, are routed through the same delay queue to ensure that inbound and outbound packets correctly interfere with one another. The one special case is the loopback interface, which is not delayed; packets transferred via this interface bypass the delay queue entirely.

We have found a need to identify key points in an environment trace and take appropriate measurements when they occur. To do so, we allow replay trace records to be *tagged*. Such records are noted by the modulation layer, and timestamped. The replay layer makes use of *upcalls*, a facility we have developed, to deliver the timestamp along with that record's model parameters to interested applications. Upcalls are delivered asynchronously, much like Unix signals, but have exactly-once delivery semantics and may pass arguments and return results.

4.2. Trace formats

While primarily designed to re-create wireless environments, replay traces can be used for any target network; the only requirement is that the network over which the trace is replayed be sufficiently fast. The information carried by these trace is simple: fixed latency, per-byte latency, bandwidth, and loss rate. Traces are small, and have a header followed by a series of fixed-size entries. The model parameters in each entry are in units specified by the header.

Recorded traces capture arbitrary information about packet traffic, device characteristics, and the mobile host. We expect them to be used over many different networks and for a number of purposes. As such, they are self-descriptive and extensible. They are composed of a header followed by packet, device, and generic records. Each of the record types has a particular fixed-content section, followed by an arbitrary variable-content section. Records with the same variable format are collectively called a *track,* and their contents are fully described by a *track header* which appears before any of that track's entries in the trace.

Each entry in a recorded trace is timestamped, and has a size field and a magic number. These latter two, together with the appropriate track header, completely identify the record's contents. Fields that are constant throughout a track — for example, the protocol for all packets in a single packet track — are stored in the track header record. Other fields, such as the size of each packet, appear in each track entry record.

Packet tracks record the protocol and size of each packet, along with any number of optional items, such as sequence and acknowledgement numbers. Device tracks record information about the physical networking devices being used, one track per device. For example, we record signal-to-noise ratio, signal quality, and silence level for our WaveLAN [18] radios. Generic tracks can be used to embed arbitrary data in the performance trace; we have used them to annotate our traces with location information.

5. Status and experience

Since the ability to exchange traces between researchers is valuable, we have gone to considerable effort to document and publicize our trace formats. A description of this format is available as an informational RFC [13]. A forthcoming version of this document incorporates many valuable improvements based on feedback received from a number of researchers. We expect that this trace format will evolve into a *de facto* standard for wireless experimentation.

We have implemented support for trace modulation in the NetBSD 1.2 operating system, a variant of the 4.4 BSD Unix operating system [11]. We have used synthetic environments to evaluate a prototype of Odyssey [14], a mobile computing platform that supports application-aware adaptation. We were interested in determining the agility with which our system could react to changes in network bandwidth. To that end, we defined a set of *reference traces*, in the spirit of control systems [4], with sharp bandwidth variations. These idealized environments cannot be duplicated with real hardware, but their simplicity allowed us to fully understand the degree to which we succeeded in the design and implementation of our prototype.

We have also validated the accuracy of trace modulation with empirical traces [15]. We have not yet implemented support for hybrid traces, but plan to use it to build an accurate model of an overlay network [8] composed of WaveLAN and cellular modem. By collecting performance traces of each of these environments and combining them through hybrid trace generation, we can provide a realistic experimental testbed without the sizable infrastructure otherwise required.

6. Relationship to other approaches

Any approach to testing and evaluating a system must make tradeoffs between realism, ease of use, and reproducibility. Live experiments offer the highest degree of realism. However, wireless experiments are relatively difficult to set up and conduct, and they require a substantial infrastructure. Furthermore, for the reasons discussed earlier, it is nearly impossible to obtain reproducible results in live wireless experiments.

Simulation provides excellent reproducibility. However, it cannot hope to match the realism of live experiments due to time dilation. Since even reasonably accurate wireless models are extremely complex [1], practical realizations tend to be either too simplistic or too computationally expensive. For example, Short [16], whose model is somewhat simpler than that found in [1], explicitly points out the sluggishness inherent in mobile simulation; attempts at speeding up the simulation have yielded only modest improvements [9].

Trace modulation provides an attractive balance. It comes close to the realism of live experiments for two reasons: first, all the system and application layers above the lowest network layer are executed exactly as they would be in live use; second, time is real rather than simulated. However, reproducibility under network modulation is substantially better than under live experiments; the variation in network quality is driven by a trace rather than a capricious infrastructure. Finally, experiments with trace modulation are no more difficult to carry out than controlled experiments on an isolated LAN.

Of course, trace modulation is not a panacea. A single trace can only capture one point in a widely varying space. Since processes that use the network often exhibit some degree of nondeterminism, even trace modulation cannot offer precisely reproducible results. Since the lowest networking layers are emulated, subtle phenomena involving these layers will not be accurately captured by trace modulation. Hence, trace modulation should not be viewed as a complete replacement for live testing and evaluation. Rather, it should be viewed as a complementary technique that substantially reduces the need for live experiments, but provides a more responsive, realistic environment than that arising from emulation. While trace modulation is especially valuable in the context of mobile computing, it can be used in testing and evaluating any complex distributed system.

Individual components of our methodology have been used before in isolation. For example, IP packet tracing for analysis and debugging using the Berkeley Packet Filter [10] and `tcpdump` [17] is common in the Internet networking community. Network emulation software of various kinds ranging from fault injection tools [3] to centralized emulators [2] are commonplace. The key insight of our approach is that combining these individually simple ideas can result in an overall methodology of immense value in building mobile systems.

7. Conclusion

Trace modulation arose as a solution to three thorny problems in mobile computing: the need to isolate application and network complexity, the need to subject mobile clients to networking conditions that are realistic yet reproducible, and the need to explore new wireless infrastructures and mobility patterns. Our experience to date suggests that the trace-based approach we advocate will continue to prove both useful and versatile.

Simple, synthetic traces may be used to explore complex software in an idealized environment. Traces exported from demanding environments enable a new mobile system to be stress-tested under real-life conditions before it is ready for deployment. A set of traces can be used as a benchmark family for evaluating and comparing the adaptive capabilities of alternative mobile system designs. Trace modulation can play an important role in debugging by deterministically reproducing the network conditions under which a bug was originally uncovered, and in exploring new infrastructures and mobility patterns. Overall, the strength of our methodology lies in its ability to combine simplicity, realism, and reproducibility — an asset that will prove invaluable in many aspects of mobile system design, implementation and evaluation.

Acknowledgments

The ideas and code described in this paper were developed in close collaboration with Randy Katz and Giao Nguyen of the University of California at Berkeley. Feedback on the trace collection format and many valuable suggestions for improvement were provided by Mary Baker, Mike Davis, Barbara Denny, David Eckhardt, Chane Fullmer, Geoff Kuenning, and Mark Lewis. Bruce Maggs and Hui Zhang helped us in designing the empirical trace modulation tools. Finally, we wish to thank our reviewers and the program committee, whose comments substantially improved this work.

This research was supported by the Air Force Materiel Command (AFMC) and the Defense Advanced Research Projects Agency (DARPA) under contract number F19628-93-C-0193. Additional support was provided by AT&T, IBM, and Intel. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFMC, ARPA, AT&T, IBM, Intel, CMU, or the U.S. Government.

References

[1] Andersen, J. B., Rappaport, T. S., Yoshida, S.
Propagation Measurements and Models for Wireless Communications Channels.
IEEE Communications Magazine 33(1):42-49, January, 1995.

[2] Davies, N., Blair, G.S., Cheverst, K., Friday, A.
A Network Emulator to Support the Development of Adaptive Applications.
In *Proceedings of the 2nd USENIX Symposium on Mobile and Location Independent Computing*. Ann Arbor, MI, April, 1995.

[3] Dawson, S., Jahanian, F.
Probing and Fault Injection of Dependable Distributed Protocols.
The Computer Journal 38(4), 1995.

[4] Dorf, R. C.
Control Systems.
In *The Electrical Engineering Handbook*. CRC Press, 1993.

[5] Duchamp, D.
Issues in Wireless Mobile Computing.
In *Proceedings of the Third Workshop on Workstation Operating Systems*. Key Biscayne, FL, April, 1992.

[6] Eckhardt, D., Steenkiste, P.
Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network.
In *Proceedings of the ACM SIGCOMM96 Conference*. Stanford University, CA, August, 1996.

[7] Forman, G. H., Zahorjan, J.
The Challenges of Mobile Computing.
IEEE Computer 27(4), April, 1994.

[8] Katz, R. H., Brewer, E. A.
The Case for Wireless Overlay Networks.
In *SPIE Multimedia and Networking Conference*. January, 1996.

[9] Liu, W., Chiang, C.-C., Wu, H.-K., Jha, V., Gerla, M., Bragrodia, R.
Parallel Simulation Environment for Mobile Wireless Networks.
In *Proceedings of the Winter Simulation Conference*. 1996.

[10] McCanne, S., Jacobson, V.
The BSD Packet Filter: A New Architecture for User-level Packet Capture.
In *Proceedings of the 1993 Winter USENIX Technical Conference*. San Deigo, CA, January, 1993.

[11] McKusick, M.K., Bostic, K., Karels, M.J., Quarterman, J.S.
The Design and Implementation of the 4.4BSD Operating System.
Addison Wesley, 1996.

[12] Nguyen, G.T., Katz, R.H., Noble, B., Satyanarayanan, M.
A Trace-Based Approach for Modelling Wireless Channel Behavior.
In *Proceedings of the Winter Simulation Conference*. 1996.

[13] Noble, B., Nguyen, G., Satyanarayanan, M., Katz, R.
Mobile Network Tracing.
October, 1996.
RFC 2041.

[14] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., Walker, K. R.
Agile Application-Aware Adaptation for Mobility.
In *submitted for publication*. 1997.

[15] Noble, B., Satyanarayanan, M., Nguyen, G., Katz, R.
Trace-Based Mobile Network Emulation.
In *submitted for publication*. 1997.

[16] Short J., Bagrodia, R., Kleinrock, L.
Mobile Wireless Network System Simulation.
Wireless Networks Journal 1(4), 1995.

[17] Jacobson, V., Leres, C., McCanne, S.
The Tcpdump Manual Page
Lawrence Berkeley Laboratory, Berkeley, CA, 1989.

[18] AT&T Global Information Solutions Company.
Architecture Specification for WaveLAN Air Interface.