# Seawind: a Wireless Network Emulator

Markku Kojo, Pasi Sarolahti, Jaakko Kyrö, and Kimmo Raatikainen
University of Helsinki, Department of Computer Science
e-mail: {markku.kojo,pasi.sarolahti,jaakko.kyro,
kimmo.raatikainen}@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/group/iwtcp/

Wide area wireless networks are being studied worldwide in order to provide better network service with good performance to customers. *The Software Emulator for Analyzing Wireless Network Data Transfers (Seawind)* is a real-time network simulator that allows researchers to measure the performance of real network applications over an emulated network that follows the user specified wireless network behavior model. Using an emulator in network performance tests has several benefits. It allows constructing and repeating specific test scenarios that are interesting to the researcher. The researcher has full control over the parameters affecting the network behavior and he can run performance tests on future networks that are not yet available for testing. By using emulator the performance tests are faster and cheaper to carry out than real field tests. A benefit of real-time emulation over simulators running in virtual time is that we can study the actual protocol implementations of different operating systems and applications.

Seawind emulator allows examination of data transfers over wireless networks such as GSM and GPRS. It can be used as a tool for network tuning, optimizing transport/application protocol performance, and studying possible feature interactions between the network behavior and transport/application layer protocols. So far Seawind has been used in a number of TCP performance studies. It was also used in the Monads demonstration in MobiCom 2000.

Seawind is designed for studying the data transfer of a single user. The information obtained with the emulator can be used to understand how the data transfer of a single user is affected by different (wireless) network characteristics, like delay and packet loss, and the competing traffic of other users. The behavior of a subnetwork to be emulated is defined with the Seawind parameter sets containing about 30 parameters for adjusting the network behavior. The actual emulation is performed in a Seawind *Simulation Process (SP)*. When SP receives a user data packet, it emulates the underlying network behavior according to the current parameter settings and thus affects the delivery of the user packet giving the user an impression that the data connection passes through a (wireless) link with given characteristics.

Seawind captures the data packets from the network device driver, e.g. from *Point-to-Point Protocol (PPP)* output, thus behavior of any network, transport and application protocols can be studied transparently to the protocol implementations. Seawind collects log of every arriving packet and the simulation events related to the packet. For example, a tcpdump-compatible output is provided for IP packets in order to allow use of existing tools for analysis. When tcpdump output is combined with Seawind event log, the network behavior and its impact on protocol performance can be analyzed in detail.

Seawind is implemented entirely in user-space, which makes it possible to port it easily to different Unix platforms. It provides a graphical user interface for specifying different test scenarios and saving the parameters for future use. The user may also specify the group of applications that is used for generating the workload in the tests. Sequence of test runs can be given to be executed automatically. Optionally, the user can use Seawind to construct the emulated wireless link and use any application manually for communicating over the wireless link. The packets can also be routed to and from the Internet at the either edge of the emulated network. External hosts can be attached to Seawind by a serial cable, allowing any system to generate the network data over Seawind.

The most challenging implementation issue of Seawind is to emulate the network behavior accurately in real-time on top of a non-realtime operating system (e.g. Linux). Obviously, hard real-time is impossible to achieve, but it is necessary that the Seawind events are triggered reasonably close to the scheduled time. In our experience the accuracy of Seawind remains within 1 ms with rare exceptions. This accuracy is sufficient for emulating links with bandwidth of few hundred kilobits per second. The event timestamps are recorded in order to evaluate the accuracy of emulation, and if a user-specified threshold of inaccuracy is exceeded, the user is notified. Seawind allows distribution of different components on separate network hosts connected with a high-speed LAN, which makes it possible to ensure that the time-critical parts of the emulation can be run in a lightly loaded host without competition of the critical resources.

The various additional features of Seawind include:

- **Various random distributions.** The user can use various random distributions for specifying sequence of random packet losses, different packet delays or other events. A random number sequence can be given from an external file, which allows the creation of any sequence of events.
- **Pipelining several Simulation Processes.** Multiple independent Simulation Processes can be joined together to emulate different network elements.
- **Input queue with different queue management algorithms.** Seawind can emulate an input queue with specified size and characteristics (e.g. as for router queue). The queue can be used as the basis of flow control between network elements, or the packets not fitting in the queue can be dropped by using specified queue management algorithm, e.g. drop-tail or *Random Early Detection (RED)*. In addition, *Explicit Congestion Notification (ECN)* is implemented in Seawind.
- **Simulation state changes.** The user can specify multiple sets of network parameters that will be changed during the test run in given time intervals or as triggered by an external application. For example, with this feature we can emulate handoffs between cells with different characteristics and traffic load.
- **Modular design** to allow new algorithms and protocol filters to be easily added at any time.