Benchmark for MANET (BENCHMmanet)

User Manual

April 2005

Mohamed M. Abou El Saoud
moaz@sce.carleton.ca

Department of Systems & Computer Engineering
Carleton University

1 Required Files

The package, **`BENCHMmanet.zip`**, contains the benchmark we have developed along with some useful scripts. The files and directories included in the package are:

```
(1)   slpRunScenario.bat
(2)   slpRunAll.bat
(3)   slp_tools/slpMotion/
(4)   slp_tools/slpTraces/
(5)   slp_tools/slpAnalysis/
(6)   slp_tools/install
(7)   slp_tools/MobileModels/
(8)   slp_tools/slpSimulationSetup.bat
(9)   slp_tools/tr2Excel.bat
(10)  slp_tools/combineAnalysis.bat
(11)  slp_tools/eval.awk
(12)  slp_tools/original_common.tcl
(13)  slp_tools/cache_common.tcl
(14)  slp_tools/conf.tcl
(15)  slp_tools/event.tcl
(16)  slp_tools/rescue.tcl
(17)  slp_tools/pursuit.tcl
(18)  slp_tools/pan.tcl
(19)  slp_tools/mesh.tcl
(20)  slp_tools/vr.tcl
(21)  slp_tools/vp.tcl
```

```
(22)  slp_tools/march.tcl
(23)  slp_tools/combat.tcl
```

The files (14)-(23) contain network configuration parameters particular to each of the 10 scenarios constituting BENCHManet. In order to generate appropriate movement patterns and setup the directory structures, the script in (8) is run. This script in turn uses the implementations of the mobility models in (7) provided by [1], [2], and [3]) to generate movement files for each of the scenarios in BENCHManet; these models must be first compiled by invoking (6). The movement files generated are stored in (3). The core simulation drivers are found in (12) and (13); the former is used when the original SLPManet is desired, while the later is used for the modified SLPManet which supports extended caching. NS-2 output traces are stored in (4). Script (11) analyzes trace files. To combine the trace analysis of all tests in a benchmark run, the script in (9) is invoked. To combine the results of several benchmark runs, the script in (10) is invoked. Analysis results are stored in (5).

The scripts in (1) and (2) automate the whole simulation and analysis process; the first automates running the procedure of simulating a particular scenario test, while the latter automates running the entire benchmark.

2 Installation Steps

1. Copy the package **`BENCHManet.zip`**, to **`ns-2.27/`**.

2. Unpack the package: **`unzip BENCHManet.zip`**.

3. Compile the mobility models by typing **`./install`** under **`slp_tools/`**.

3 Running Simulations

3.1 To Run a Particular Scenario

Run the script:

`./slpRunScenario.bat scenario runDirectory seed <-cache>`

The script automates the process of simulating `scenario`. `runDirectory` is the name of the sample. `seed` is used to set the seed of the random number generators. If the `-cache` option is specified, the modified version of SLPManet that allows extended caching is used instead of the original SLPManet. The script in turn invokes other scripts to set up directory structures, generate new movement files (if not already existing) and run the simulation for the given scenario.

For instance, in order to run the conference scenario with the original SLPManet, setting a seed of 2, one would type:

```
./slpRunScenario.bat conf run0 2 -cache
```

The movement file will be stored in:

```
slp_tools/slpMotion/run0/conf.motion
```

The trace file will be stored in:

```
slp_tools/slpTraces/run0/original/conf.tr
```

To analyze the generated output trace file, run:

```
awk -f slp_tools/eval.awk slp_tools/slpTraces/run0/original/conf.tr
```

3.2 To Run the Entire Benchmark

Run the script:

`./slpRunAll.bat runDirectory seed <-cache>`

The script is used to run all scenarios in the benchmark. `runDirectory` is the name of the sample run. `seed` is used to set the seed of the random number

generators. If the `-cache` option is specified, the modified version of SLPManet that allows extended caching is used instead of the original SLPManet. The script in turn invokes other scripts to setup directory structures, generate new movement files (if not already existing), run the simulations, and analyze trace files.

For instance, in order to run the benchmark, with a seed of 2, and enabling caching, one would type:

```
./slpRunAll.bat runAll0 2 -cache
```

The movement files will be stored in:

`slp_tools/slpMotion/runAll0/<scenario_name>.motion`

The trace files will be stored in:

`slp_tools/slpTraces/runAll0/cache/<scenario_name>.tr`

The summary of analysis of all scenarios will be stored in:

`slp_tools/slpAnalysis/runAll0_cache.txt`

Note that the script in (8), which is invoked by `slpRunScenario.bat` and `slpRunAll.bat`, randomly generates a new set of movement files for every new `runDirectory`. If the simulation was run for an already existing `runDirectory`, then a set of movement files must have already been previously generated for this sample run, and are hence used in the simulation.

The `seed` only controls the random generators used in the NS-2 simulation. Random generators were used to randomly assign agents to nodes, and to generate random exponential inter-arrival times for user requests.

If you wish to run several samples of the benchmark, and then merge the analysis files into a single file, run:

`./combineAnalysis.bat <-cache>`

The script combines all the benchmark summary files:

`slp_tools/slpAnalysis/*_original.txt`

and produces

`slp_tools/slpAnalysis/original_results.txt`.

If the `-cache` option was specified, it will similarly merge all the cache analysis summary files and produces `cache_results.txt`.

4 Understanding the Analysis Output

The output of the trace file analysis is organized in rows, each of which represents the analysis of a simulation sample run of a given scenario. Each row contains 16 columns, whose headers are described below:

1. **`scenario`**: Name of the scenario test (e.g. **conf**).
2. **`Rqsts`**: Total number of discovery transactions.
3. **`tRplys`**: Total number of successful discovery transactions.
4. **`uRplys`**: Portion of discovery transactions satisfied by new Service Replys (SrvRplys).
5. **`Cached`**: Portion of discovery transactions satisfied through cache hits.
6. **`Succ`**: Overall Discovery Success Percentage.
7. **`minLat`**: The minimum latency consumption of all successful discovery transactions (in seconds).
8. **`maxLat`**: The maximum latency consumption of all successful discovery transactions (in seconds).
9. **`avgLat`**: The average latency consumption of all successful discovery transactions (in seconds).

10. **stdLat**: The standard deviation in latency consumption of all successful service discovery transactions (in seconds).

11. **minBW**: The minimum bandwidth consumption of all successful discovery transactions (in bytes).

12. **maxBW**: The maximum bandwidth consumption of all successful discovery transactions (in bytes).

13. **avgBW**: The average bandwidth consumption of all successful discovery transactions (in bytes).

14. **stdBW**: The standard deviation in bandwidth consumption of all successful service discovery transactions (in bytes).

15. **aggBW**: Aggregate bandwidth consumption by the entire discovery process (in bytes).

16. **wasBW**: Total wasted bandwidth due to unsuccessful discovery transactions (in bytes).

References

[1] T. Camp, J. Boleng, and V. Davies, "A survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2(5), pp. 483–502, 2002.

[2] W. Navidi and T. Camp, "Stationary Distributions for the Random Waypoint Mobility Model," *IEEE Transactions on Mobile Computing*, vol. 3(1), pp. 99–108, January/March 2004.

[3] W. Navidi, T. Camp, and N. Bauer, "Improving the Accuracy of Random Waypoint Simulations Through Steady-State Initialization," in *Proceedings of the 15th International Conference on Modeling and Simulation (MS '04)*, pp. 319–326, March 2004.