Example Simulation with the ICN Evaluator (ICE)

Mays AL-Naday

2018-03-13

Introduction

Here I will show how ICE can be used to simulate the performance of *-over-ICN technology over the Abilien graph , imported from the Internet Topology Zoo dataset (via). The Figure below shows the network topology.

Configuration

The specification of this scenario is provided in the configuration file `tools\config.yml`. The file specifies:

- 2 origins to be selected by the `Pop` policy.
- two scales of surrogates: `[2, 4]` to be placed by the `Pop` policy. This results in creating two permutations to be tested within this scenario: `{(2 origins, 2 surrogates), (2 origins, 4 surrogates)}`.
- two scales of DNS: `[2,4]` to be placed by the `Pop` policy. This increases the number of permutations to be generated for the DNS-based network to four: `{(2 origins, 2 surrogates, 2 DNS), (2 origins, 4 surrogates, 2 DNS), (2 origins, 2 surrogates, 4 DNS), (2 origins, 4 surrogates, 4 DNS)}`
- Load on the network is set to `0.4` (i.e. `40%`)
- number of random tests to be generated is set to `50`, that is each permutation of the above will have `50` psudo random selections of service points. However simulation tests have been set to `10 < 50`, as in you will simulate only for a subset of the genrated tests.

Run

```
> require(doParallel, quietly = TRUE)
> require(foreach, quietly = TRUE)
> #' configure the logging level to 'INFO' for minimum print-outs
> #' Parse the configuration file for data generation
> gcfgs <- ice::ParseConfig(config.file = "../tools/config.yml", config.env = "default")
> #' Parse the configuration file for simulation
```

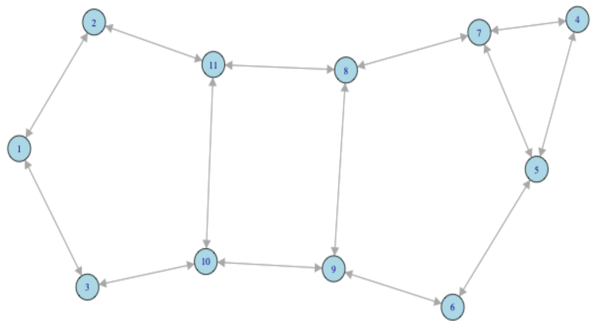

Figure 1: Abilient Topology (11 nodes, 28 edges)

```
> scfgs <- ice::ParseConfig(config.file = "../tools/config.yml", config.env = "simulate")
> #' verify input data is already generated and stored,
> #' if not then generate and save the data.
> ice::GenSimData(cfgs = gcfgs)
> #' Simulate capacity requriments for admitting input unicast traffic in ICN
> if ("icn.uc" %in% scfgs$tcfg$v) ice::SimIcnUnicast(scfgs)
#> Warning in dir.create(resdir[i], showWarnings = TRUE): '../data/results/
#> N_1000-d_zipf-params_s_0.85_ig_0-v_0.02_0.04' already exists
> #' Simulate capacity requriments for admitting input multicast traffic in ICN
> if ("icn.mc" %in% scfgs$tcfg$v) ice::SimIcnMulticast(scfgs)
> #' Simulate capacity requirments for admitting input traffic in IP
> if ("ip" %in% scfgs$tcfg$v) ice::SimIp(scfgs)
```

Process Simulation results

Now that simulation is complete, raw results are stored and hence can be processed to generate summaries

```
> icn.cap <- ice::ProcIcnUnicastCapacity(cfgs = scfgs)
> suma <- Rmisc::summarySE(data = icn.cap,
+                          measurevar = 'provisioned',
+                          groupvars = c('surrogates'))
> suma
#>   surrogates  N provisioned       sd       se        ci
#> 1          2 10   15118.380 2466.962 780.1219 1764.7584
#> 2          4 10    9507.104 1266.175 400.3998  905.7673
```

Summary

The example above can be imported into a separate script file and run from terminal as an executable.