Parallel TDC

Mojatatu Networks

Current state

- TDC Consists of more than a thousand test cases
  - It’s still growing…
  - P4TC will add 200+ cases
- Integrated into kselftests
  - Used by many downstream public/private CI services for sanity checking
    - RedHat, Linaro, etc
  - Only runs a select part of the suite
- Somewhat self contained python code
  - Only dependency to external python modules is `scapy`
- Painfully serial way to solve a embarrassingly parallel workload
  - Takes a long time to finish on some debug configs
  - Why we are not using all these spare cores to do some useful work?

Current state -next

- Every test definition is a self contained unit if using net namespaces
  - No test should interfere with each other
- Split the test run into 3 phases
  - Setup all tests resources, run all tests, report results
- Run tests over a worker pool
  - Resizable with the `-J` argument
  - Implemented using Python’s own thread pool library
- Total runtime dropped close to 2.5x in full run
  - Even though the tests serialize over rtnl_lock()

Upcoming changes

- Still fixing/improving some rough edges on the CI services
  - In contact with the tuxsuite guys to make tdc as green as possible
- Let the bulls run
  - Make kselftests run the entire suite by default
- Improve handling of resources so rcu and workqueue stops trolling us
  - Easy way out is to just `sleep 2` everywhere but it’s a massive workaround
  - Maybe some fancy netlink notification callback scheme?
- Improve resource setup time
  - We rely heavily on iproute2 but debug configs make fork() pretty slow
  - As this is the same for every test perhaps calling netlink directly is better

Thank you!