Qemunet: an Approach to an Automated Virtualized Testbed

Pavel Boyko Andrey Mazo

Institute for Information Transmission Problems

Workshop on ns-3, March 2011

Missing features in existing tools

Use case in mind

- Using network testbeds for testing distributed software

- Inject some event $\longrightarrow$ get system response
- Measure not steady-state but transient characteristics
- Multi-step experiments
- Branching in experiments through simple run-time analysis

Per-command synchronization

- Multiple command processing units
- Single command scheduling/synchronizing unit

Task	related commands for different processing units
Commands	Unix shell non-interactive commands
Sending commands	via system console

Why virtualize?

Real computers are hard to deal with

- Hard to obtain sufficient quantity
- Hard to physically interconnect
- Hard to manage (power on, power off, update, ...)

Full virtualization or not?

+ Run software unmodified
+ Not bound to existing hardware
— Performance

Integration into ns-3

Network simulator

- High-fidelity channel and device models
- Fully simulated (non-VM based) nodes
- Topology and network state knowledge

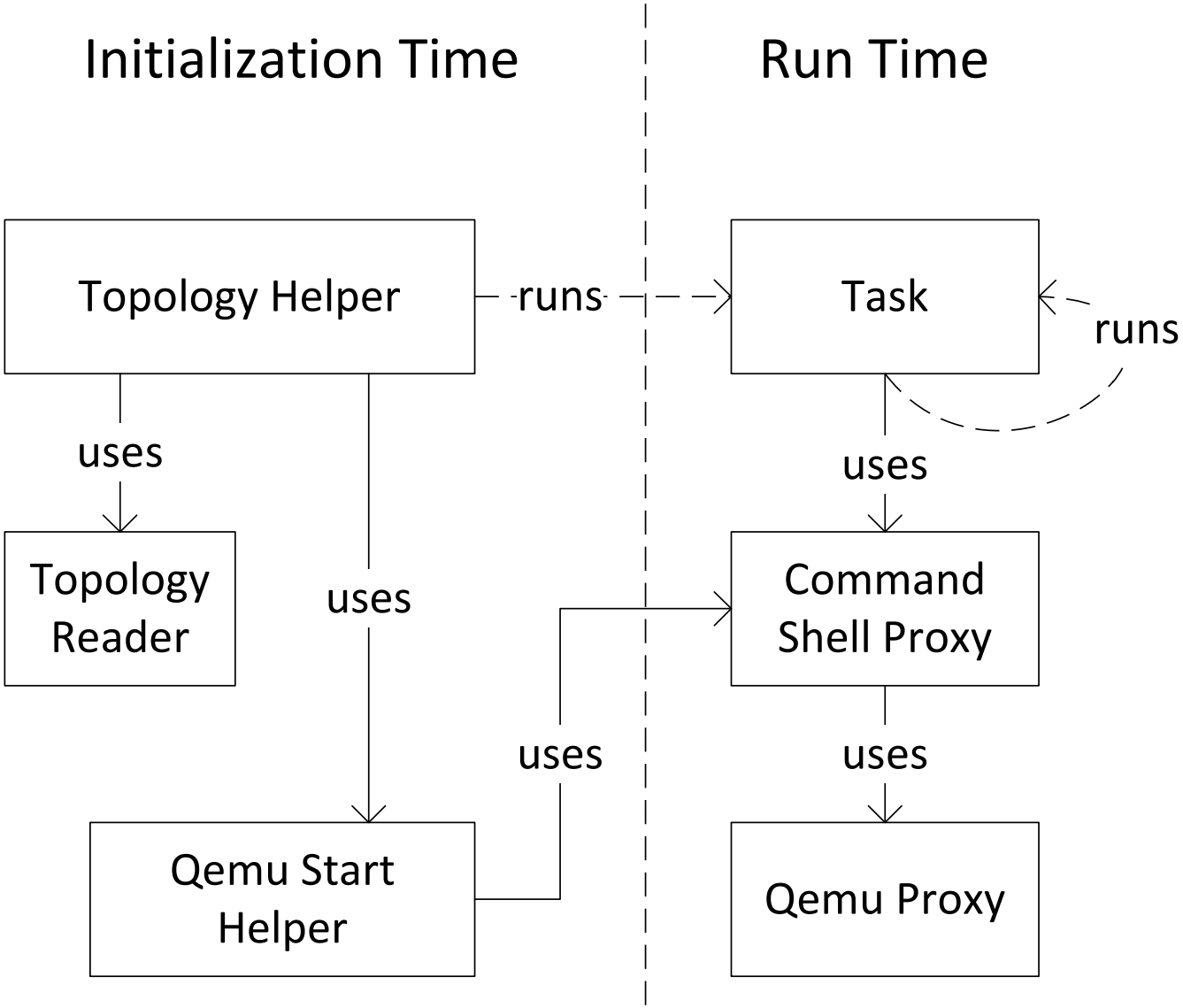
Command synchronizer

- Sync executed commands (in VMs) with in-simulator events
- Sync executed commands (in VMs)

Testbed configurator

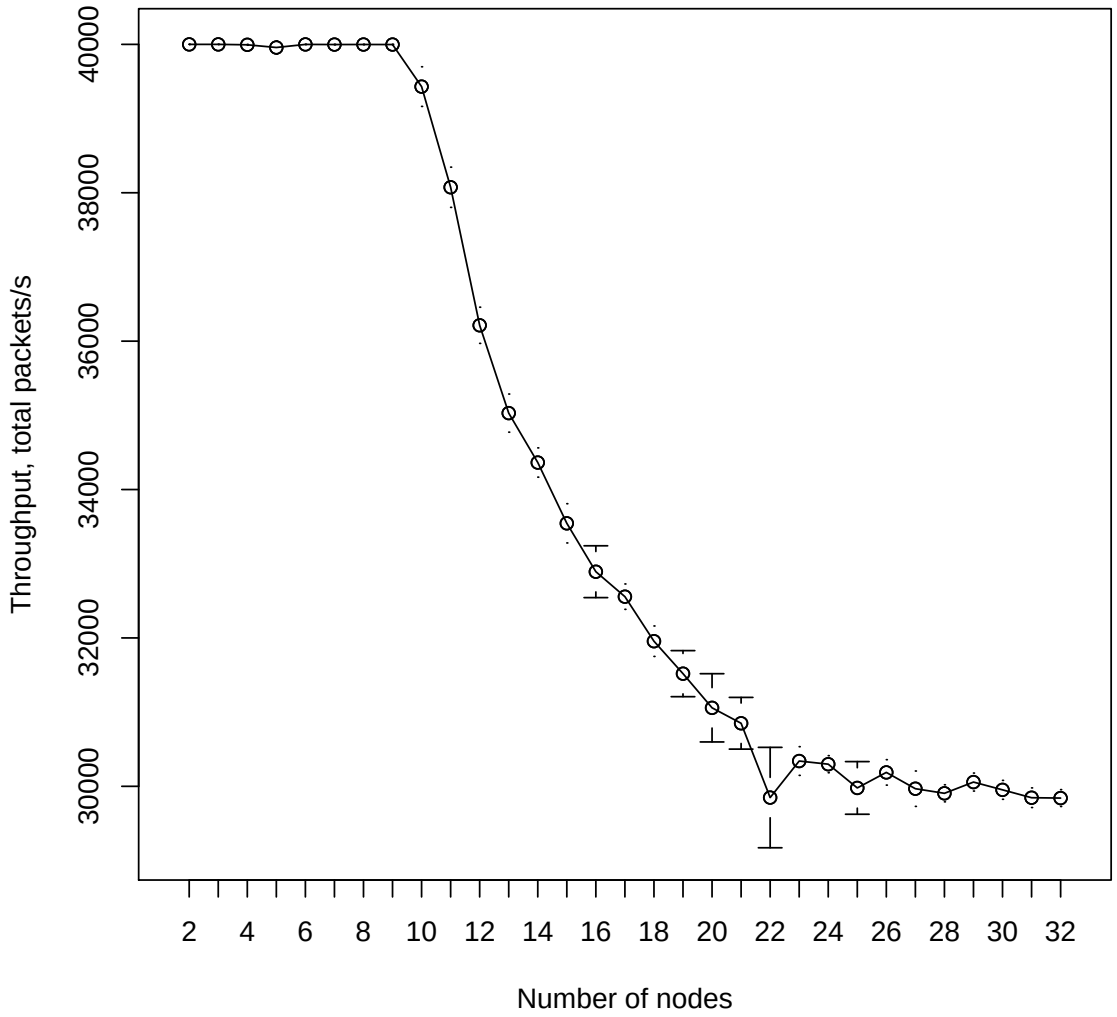
- Launch of VMs
- Post-boot configuration in VMs
- Interconnection of VMs

Qemunet architecture



Other components: NodeType, NodeDescription, InterfaceDescription, LinkDescription, ChannelDescription

Scalability



- Machine: 2.27GHz x 8 x HTT, 12GB RAM
- Max total pckts/s: 40k
- CORE's max total pckts/s (scaled): 400k
- Max voice streams: 80
- Max video streams: 15

Conclusion

Ns-3

- A network simulator
- Wall clock for synchronization
- Manage external processes