**System Architecture**

**Clients**
- Each runs multiple real applications.

**ModelNet Emulators**
- Each emulates entire topology.
- Moves packet by reference.
- Multiple emulators to increase bandwidth.
- Distributed Rate Limiting (DRL) to reduce intra-emulator traffic.

**Problem:** Vendors tend to implement OpenFlow differently. How can we capture and emulate such variations?

**Switch Replication**
- Hardware idiosyncrasies do matter. How to let OVS capture them?

**Inside Emulator**
- **Virtual Link** - Imposes bandwidth, delay and loss on packets.
- **Dummy Node** - Packets routed via shortest path.
- **User Node** - Packets follow user defined behaviors.
- **High-Fidelity Switch Replicas**

**Clients**
- Offers unique IP addresses

**Emulators**
- Application Traffic

**Switch**
- Synthetic Traffic

**DRL Control**

**Virtual Topology**
- User
- 1 Gbps
- Dummy
- 1 Gbps

**Problem:** How can we squeeze more bandwidth out of a single emulator host?

**Graph 1**
- Bandwidth vs Number of Hops
- 30 flows, 1Gbps/unlimited links, OVS/Shortest Path

**Graph 2**
- Bandwidth vs Number of Hops
- 30 flows, 1Gbps/unlimited links, OVS/Shortest Path

**Graph 3**
- Distribution of Redis Request Completion Times under Different Switches

**[1] Profiling**
- Quantify idiosyncrasies in hardware OpenFlow switch.

**[2] Modifying OVS and Verify**
- Replicate hardware idiosyncrasies in OVS and compare performance.