Network administration is complicated and error-prone.

- Code in many languages runs on diverse hardware.
- Policy enforcement must scale over large networks.
- Tenants have diverse needs and usage patterns.

Networks lack unified abstractions.

- Target a single type of device or platform.
- Assume a single point of control.
- Disconnect between intended policies and actual implementation mechanisms. For example: “Ensure that all web traffic traverses a firewall” vs. “Match HTTP traffic & forward out switch port 4”

Merlin: Specify policies in a high-level language that is compiled to low-level code for network devices.

Policies may be:
- **Partitioned** into components for network devices,
- **Distributed** for finer grained enforcement,
- **Safely delegated** to tenants for network federation.

Examples

**Bandwidth Guarantees**

```plaintext
{ ipSrc = 10.1.1.1 and
  ipDest = 10.1.1.2 and
  ipProto = 6
  and tcpDst = 80 } -> .+ nat .+ dpi .+ at min(1Gb/s)
```

**Flexible Defense-in-Depth**

```plaintext
forall true
  -> (.+ fire1 .+ fire2 .+ ) |
  (.+ fire2 .+ fire1 .+ )
```

**IP Multicast Control**

```plaintext
forall ( ipSrc = 224.0.0.1 or
  ipSrc = 224.0.0.2 or
  ipSrc = 224.0.0.3 )
  -> compress .+
  at max(16Gb/s)
```

**Resource Isolation**

```plaintext
forall ( ipSrc = 10.1.1.1/8 )
  -> .+ m1 .+
forall ( ipSrc = 10.1.1.8 )
  -> f(+ m1 .+)
```

**QoS Guarantee**

Merlin reserves 90% bandwidth for Hadoop job reducing effect of background traffic