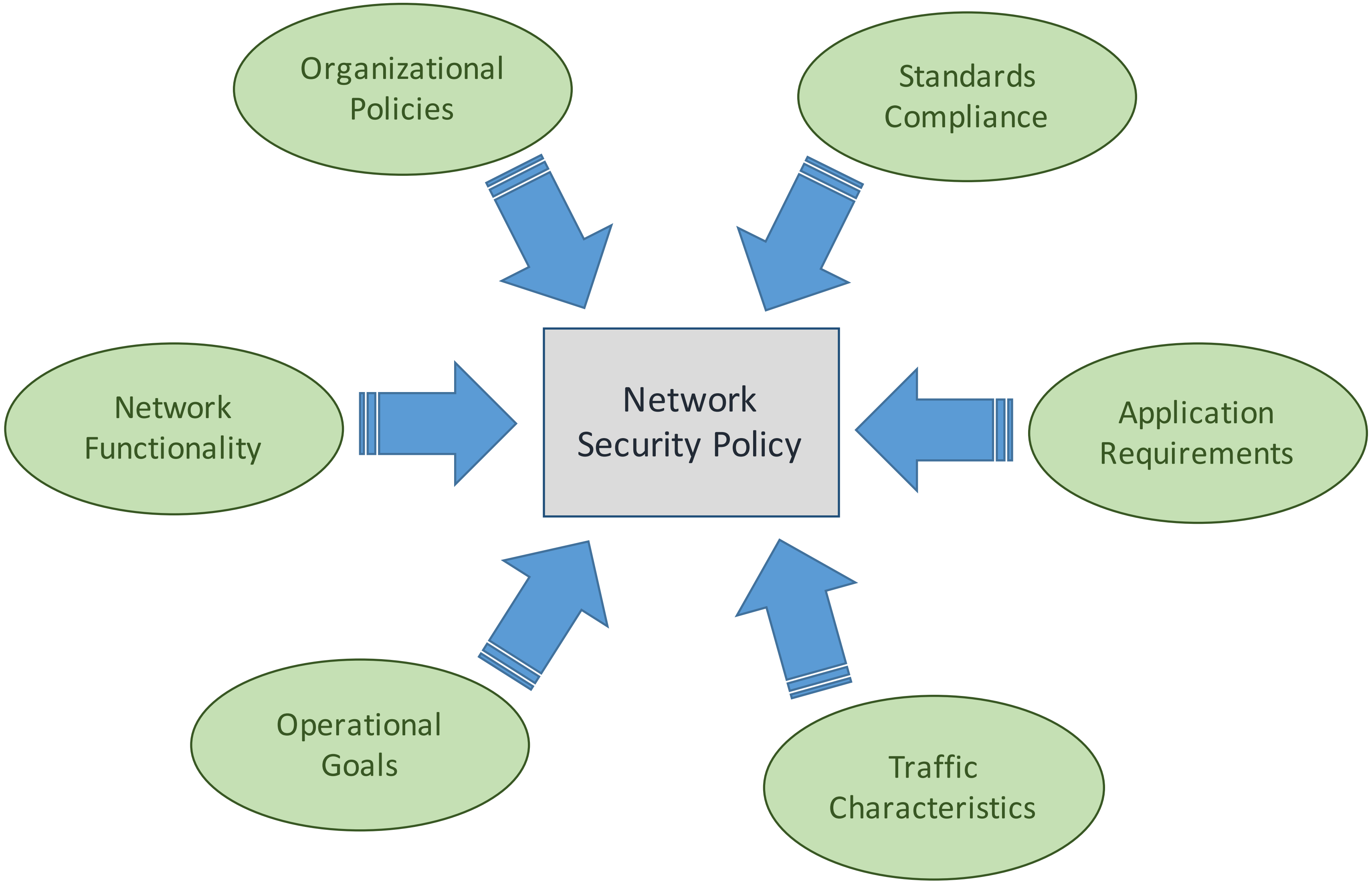
Oreo: Transparent Optimization to Enable Flexible Policy Enforcement in Software Defined Networks

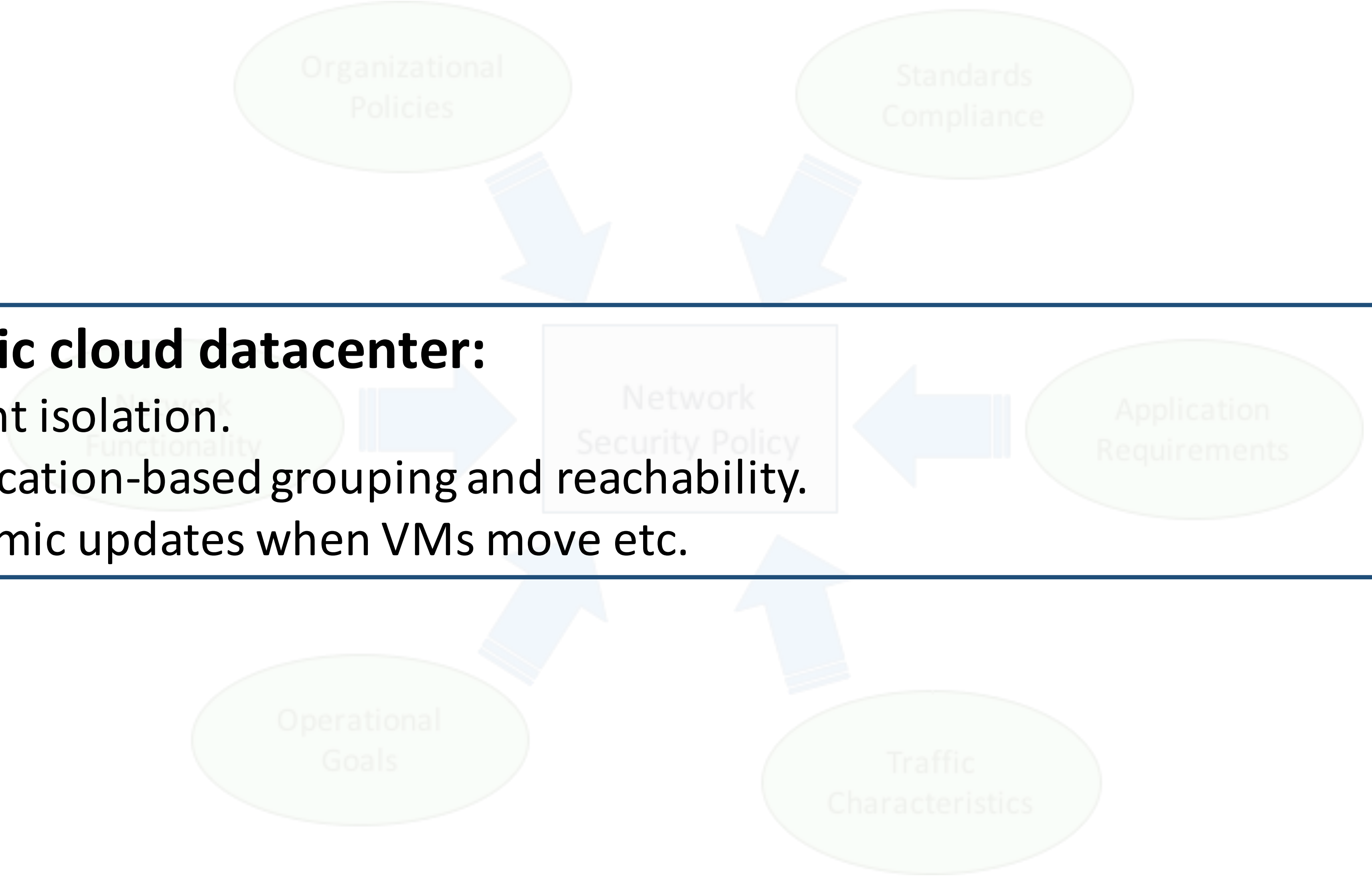
Santhosh Prabhu, Mo Dong, Tong Meng, Phillip Brighten Godfrey, Matthew Caesar

University of Illinois at Urbana Champaign

Supported in part by NSA-SOS Network Hypothesis Testing Lablet

Network Policy Overview





Public cloud datacenter:

Tenant isolation.

Application-based grouping and reachability.

Dynamic updates when VMs move etc.

Large financial/healthcare organization:

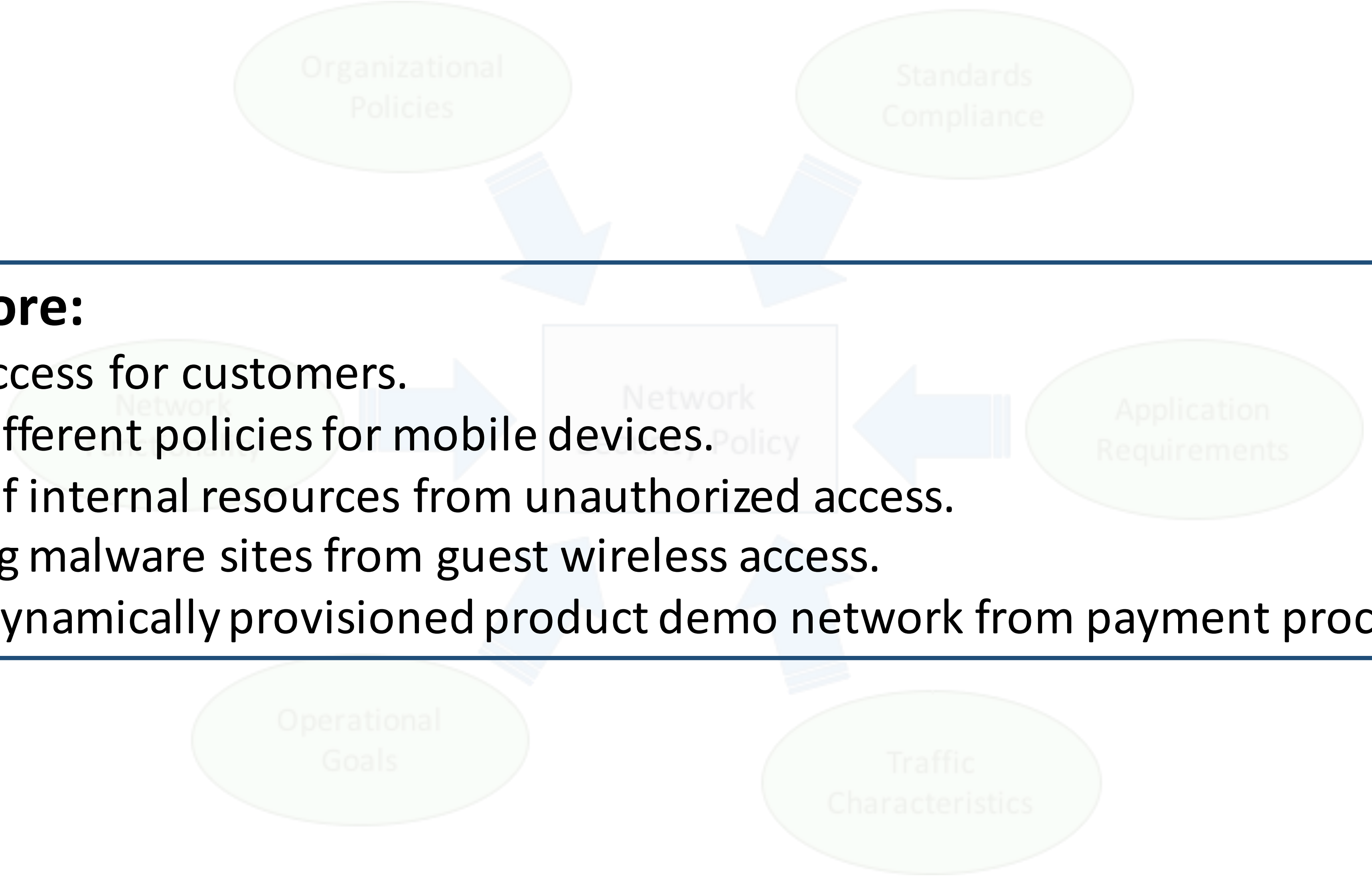
Isolation between departments.

Policies for BYOD.

Protection from external threats.

Different departments use different Internet gateways.

Different departments goes through different firewall and IDS groups.



Retail store:

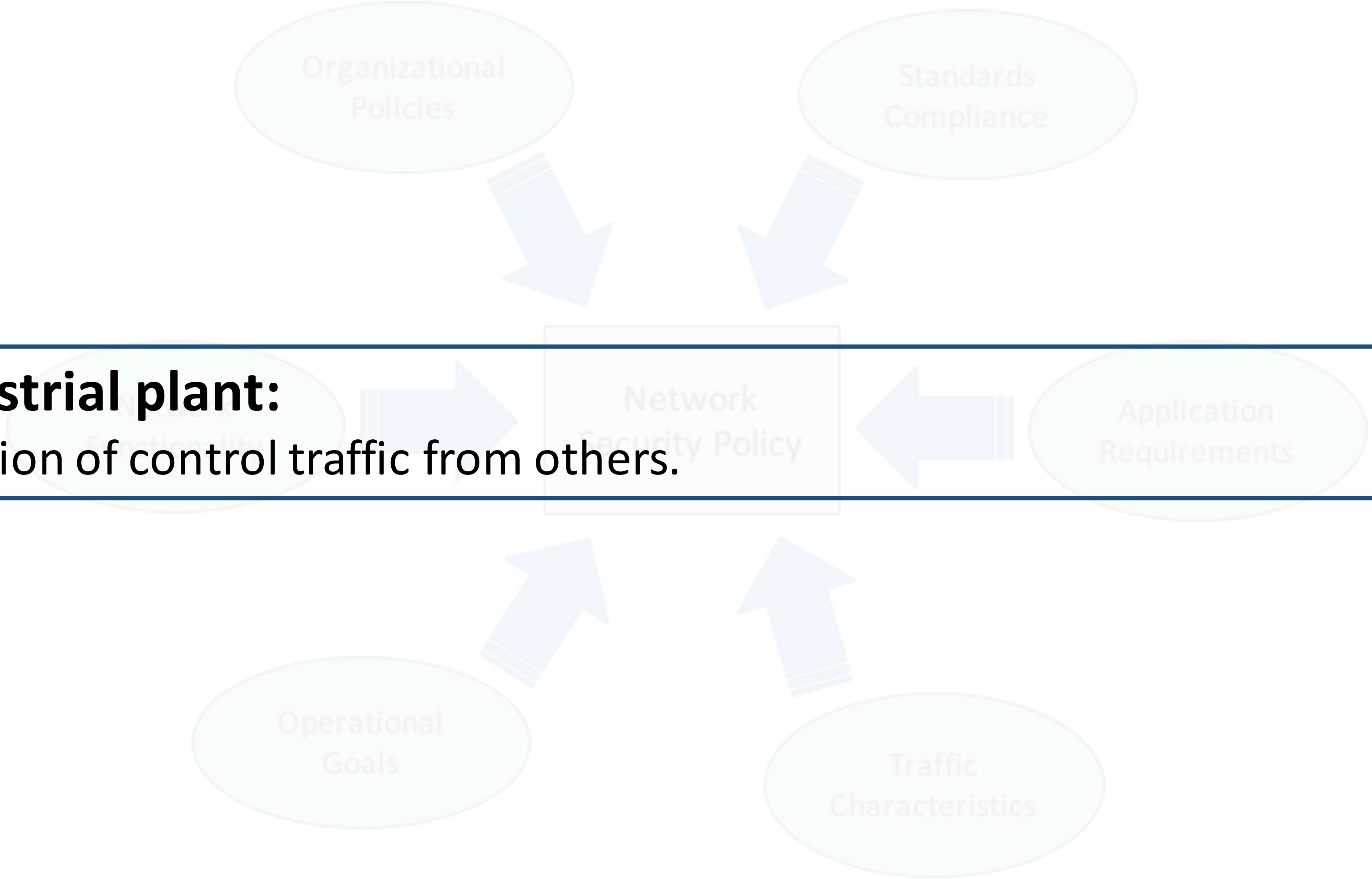
Internet access for customers.

Possibly different policies for mobile devices.

Isolation of internal resources from unauthorized access.

Blacklisting malware sites from guest wireless access.

Isolating dynamically provisioned product demo network from payment processing network.

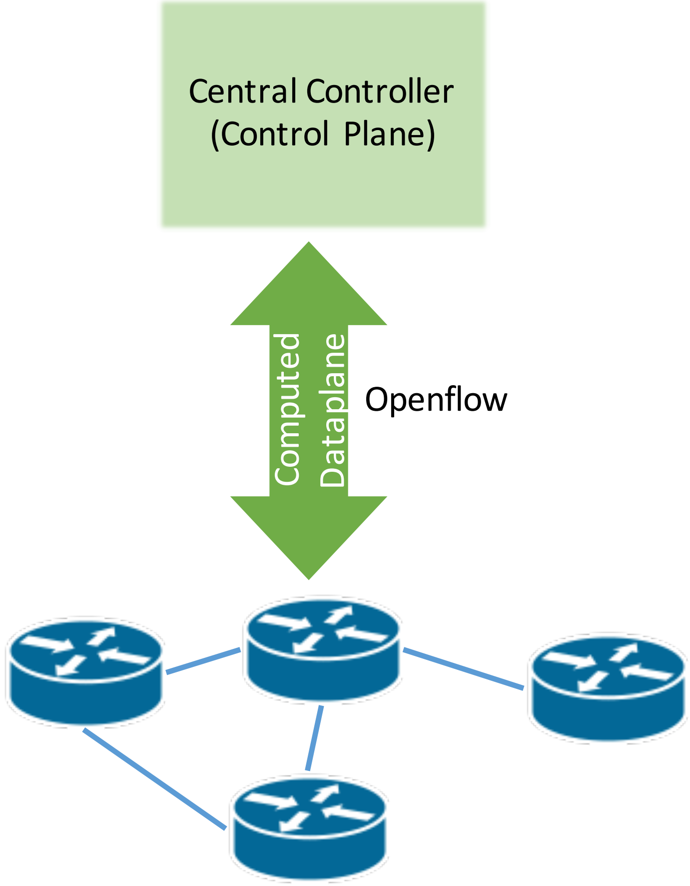


Industrial plant:

Isolation of control traffic from others.

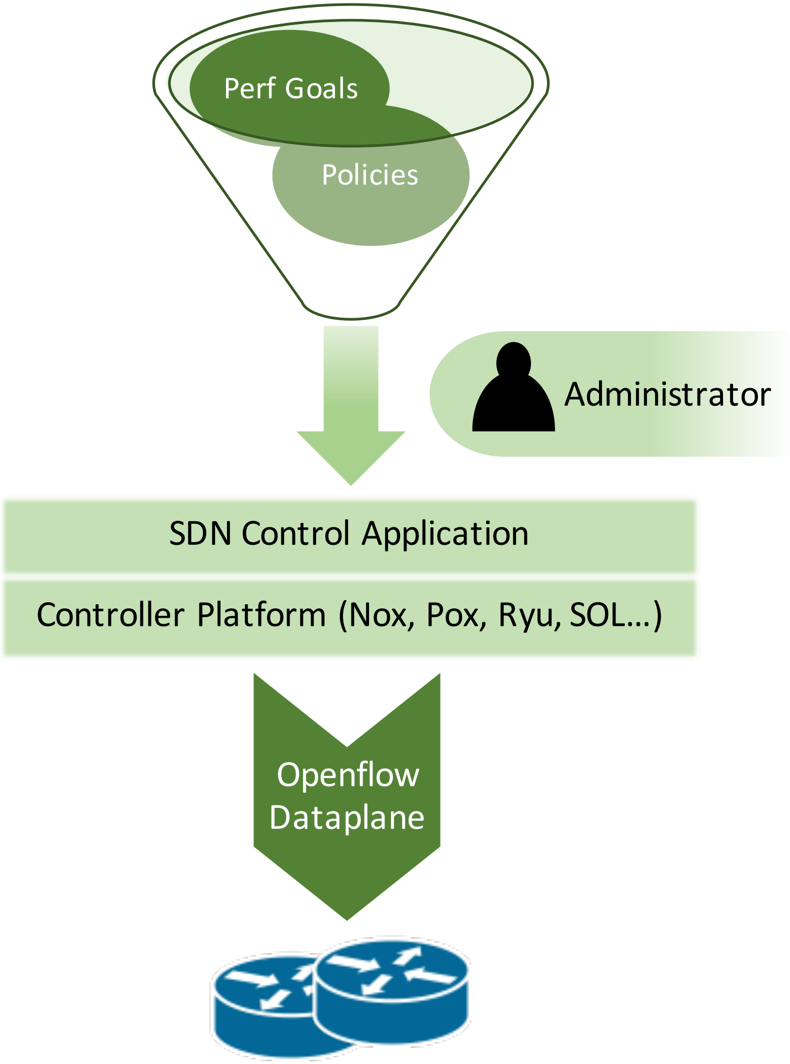
Performance, on the other hand, is well understood. Operators use protocols like MPLS-TE, Fast Reroute, OSPF (Shortest Path Routing) etc., which require only minimal configuration.

Software Defined Networking – A (very) brief overview



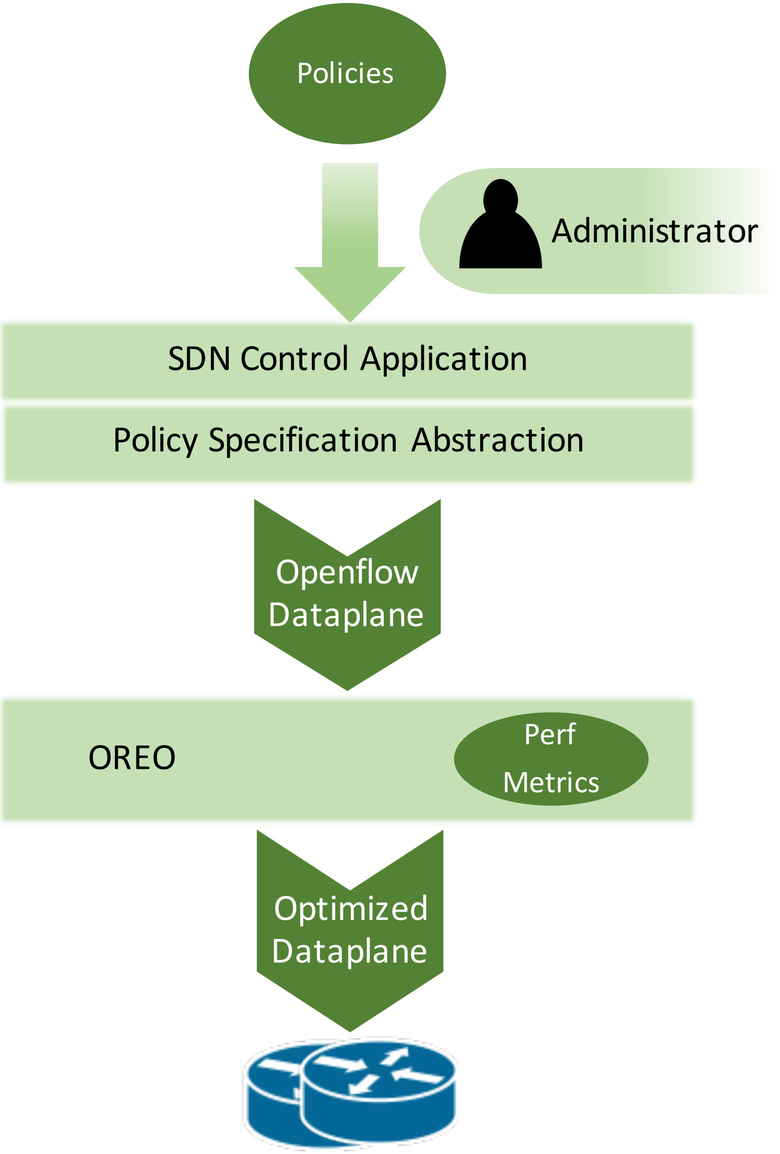
- Proposed in 2008 as a completely new paradigm in networking.
- Uses a centralized control plane (as opposed to distributed protocols in traditional networks).
- Rules are computed and pushed to the switches (known as the dataplane).
- Openflow - Open standard for communication between switches and controller
- Promise of revolutionizing networking, including policy management.

Policy Enforcement with SDN



- Tight coupling of policy enforcement and performance optimization
- Controller platforms present a combined abstraction for both
- Makes policy enforcement harder
- Performance should ideally not require administrator involvement

Oreo: Transparent performance optimization in SDNs



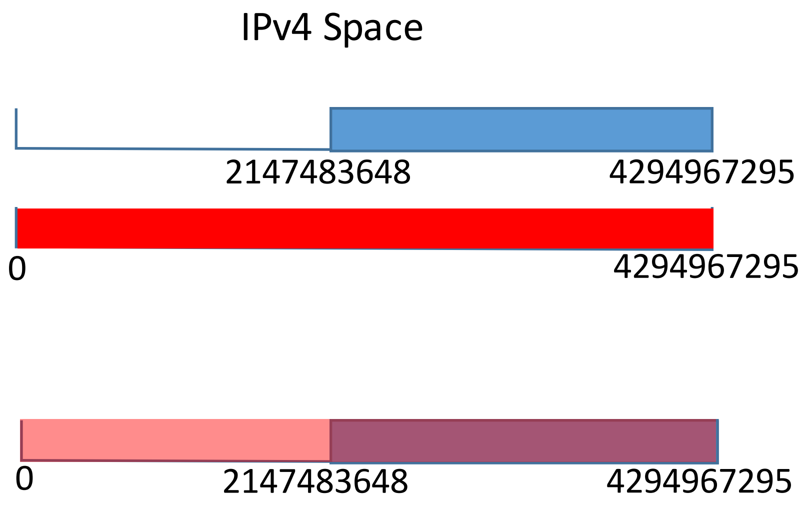
- Model-based optimization
- Network-wide dataplane model describes each flow
- Optimization retains end-to-end reachability characteristics (does not violate the policies enforced by the controller)
- Use well-defined optimization objectives, done automatically:
  - Reduce path stretch
  - Reduce switch memory usage
  - Static/dynamic traffic engineering
  - …

Optimization – A programming analogy

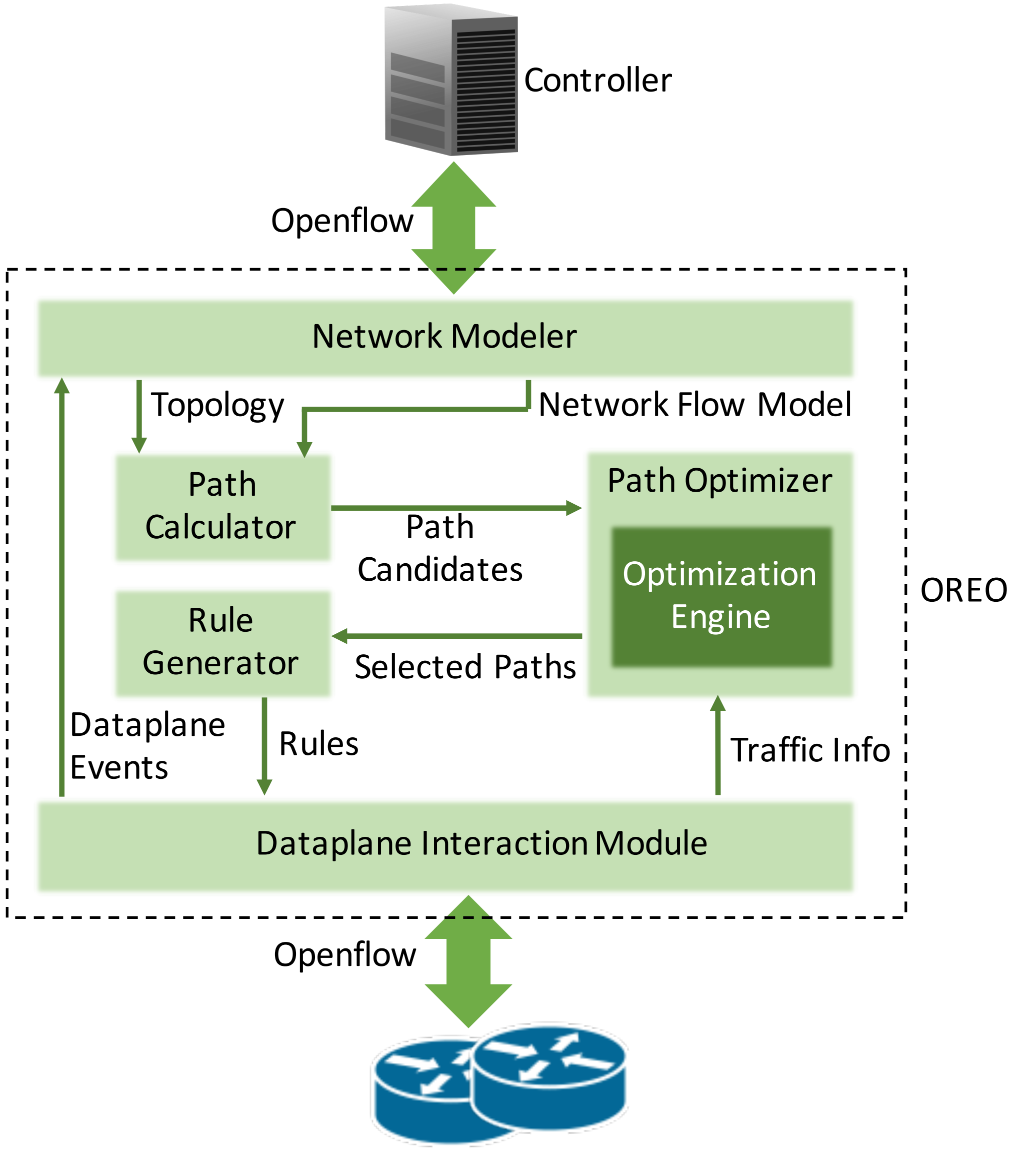
Compilers	Network Optimization
Transparently optimize programs	Transparently optimize network dataplane
Applies to wide variety of programs	Applies to all kinds of policy enforcement applications in the control plane
Optimization of single file/translation unit	Optimization of rules at one device (Done by existing work)
Link-Time Optimization	Network-wide Optimization (Oreo)
E.g. Dead code elimination	E.g. Unreachable rule elimination

Traffic classes: Key to dataplane optimization

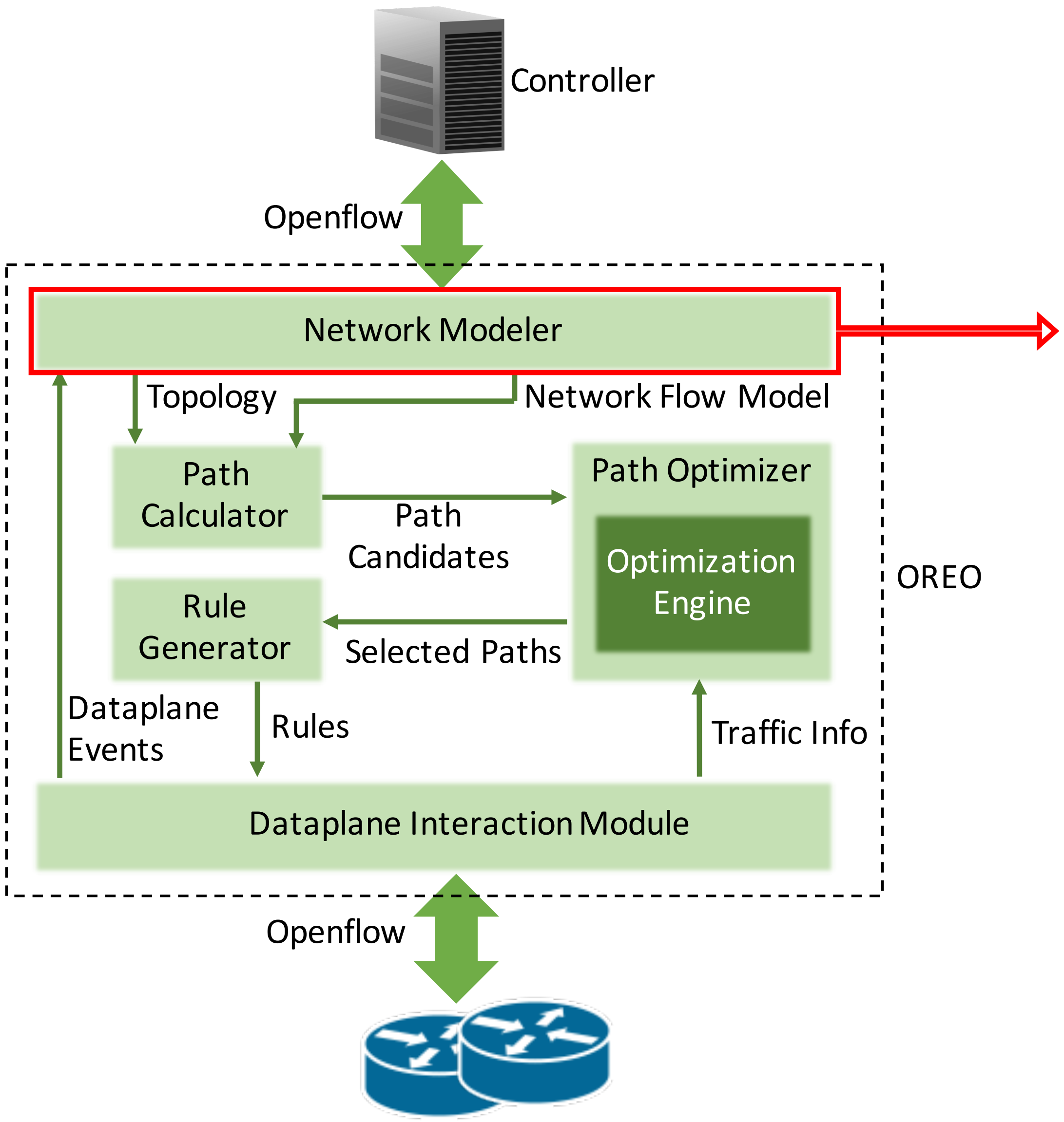
- Traffic categorized into equivalence classes.
- Each EC is a set of packets
- Two packets in the same EC will have the same behavior in the entire network.
- Example:
  - Rule 1 matching on 128.0.0/1, priority 1
  - Rule 2 matching on 0.0.0.0/0, priority 2
- Equivalence classes:
  - 0 – 2147483647 (Packets that match Rule 1)
  - 2147483648 – 4294967295 (Packets that match Rule 2 but not Rule 1)
- Redefining paths for one equivalence class doesn‘t affect another.



Architecture

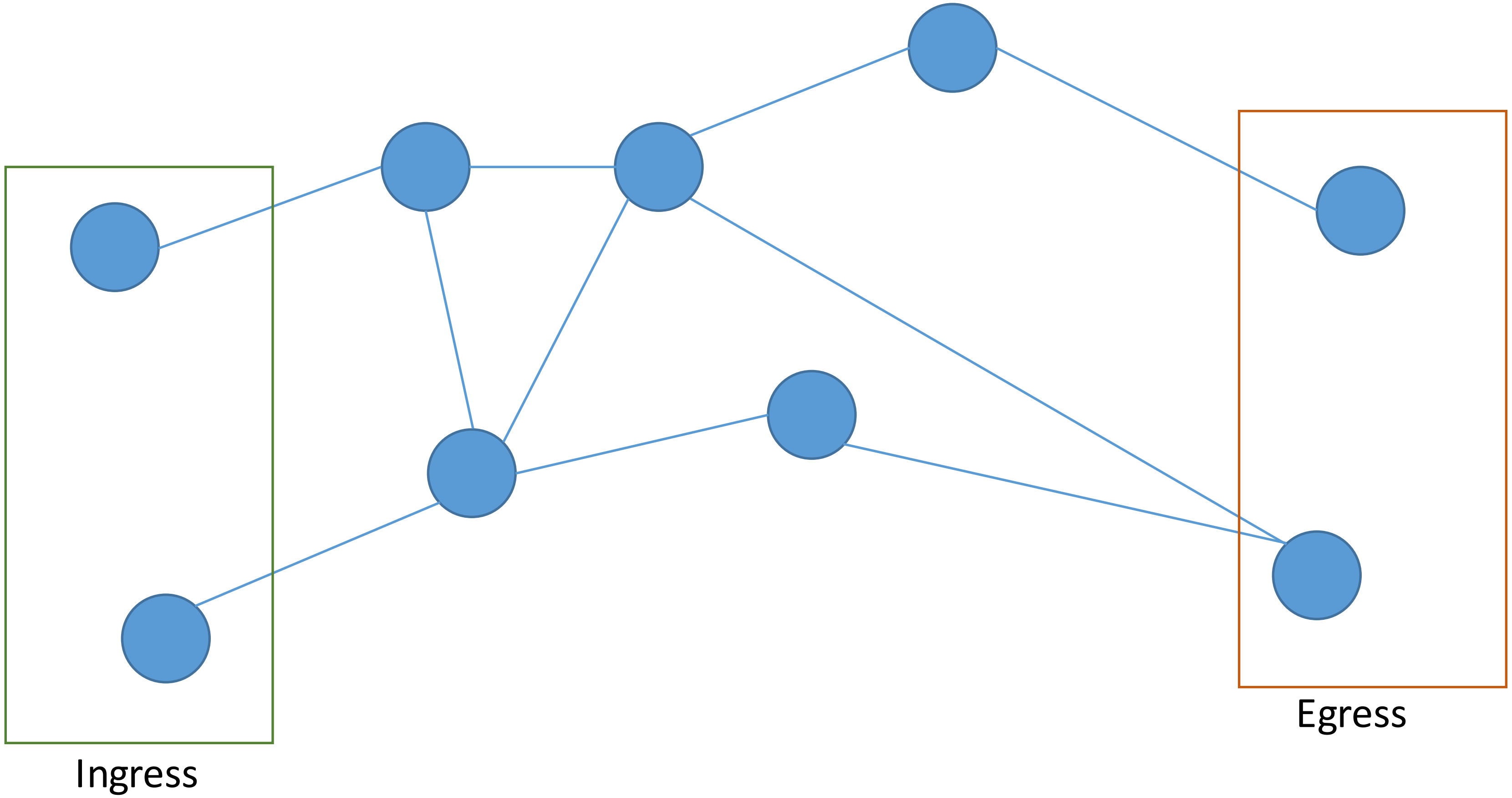


Architecture

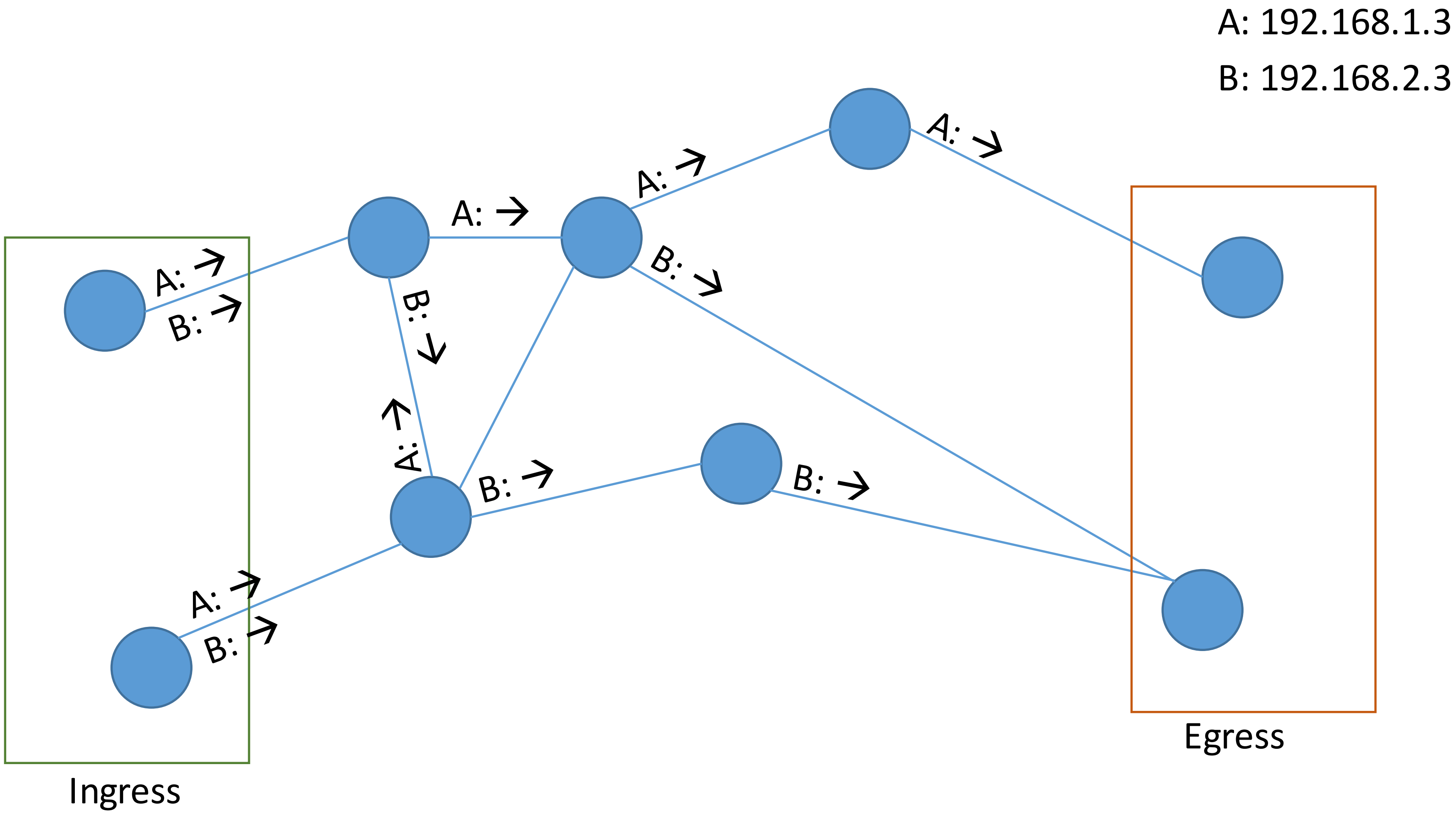


- Computes equivalence classes of traffic, which could be manipulated independently of each other.
- For each equivalence class, determines end-to-end reachability information, and paths traversed.
- Current experiments use *Veriflow* dataplane verifier.
- Veriflow computes equivalence classes of traffic for the purpose of verification.

Example: Topology

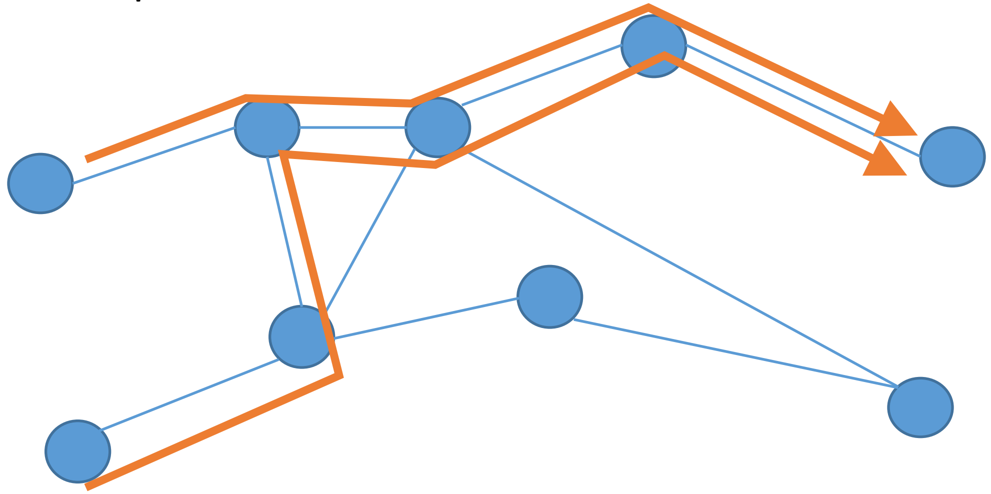


Example: Dataplane

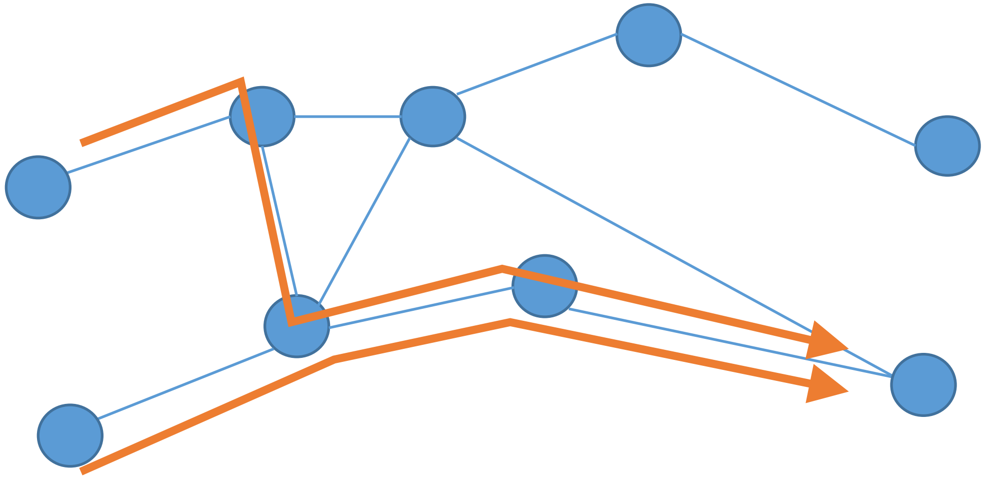


Example: Flow Model

IPv4 Equivalence Class: 3232235779 - 3232235779

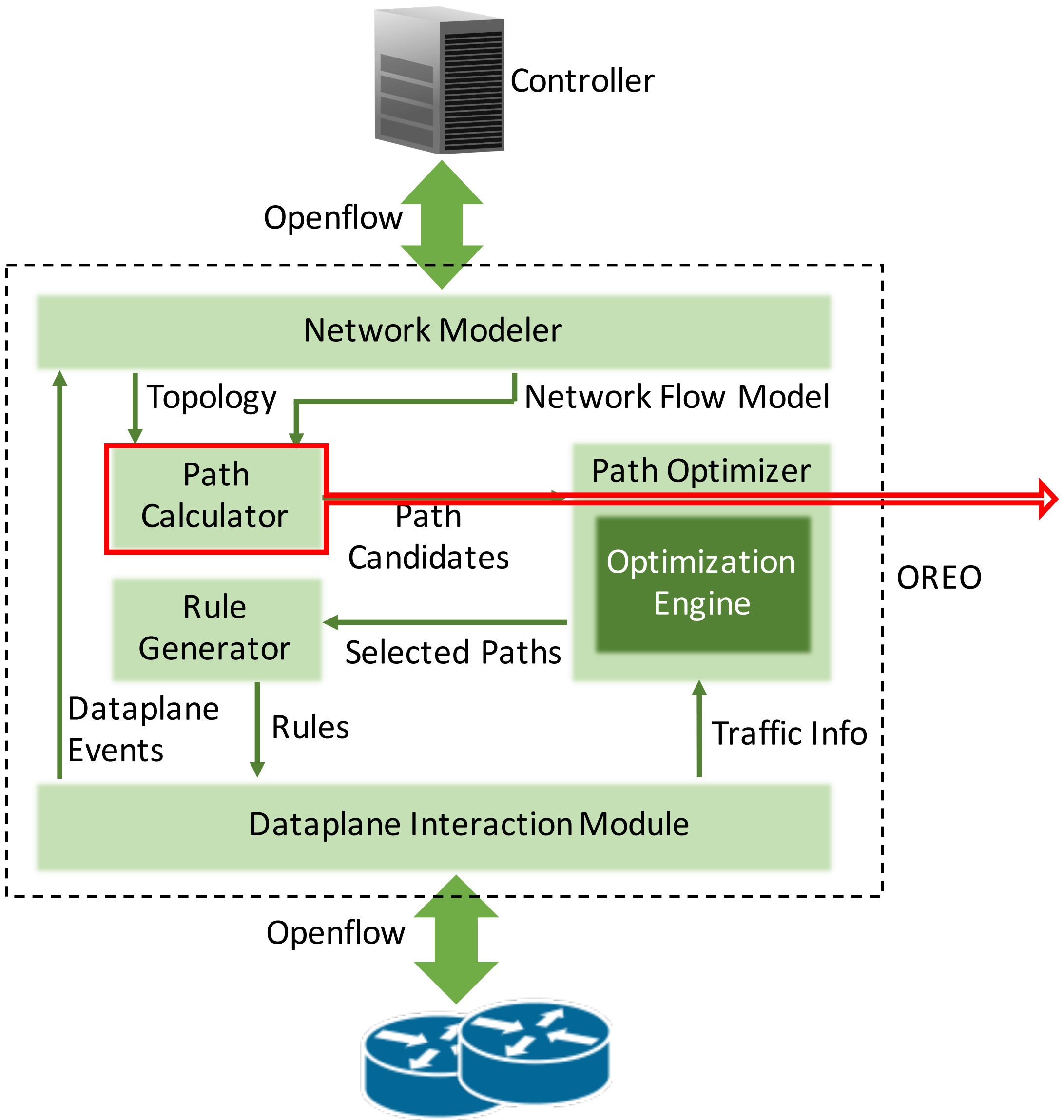


IPv4 Equivalence Class: 3232236035 - 3232236035



Other Equivalence Classes: Drop

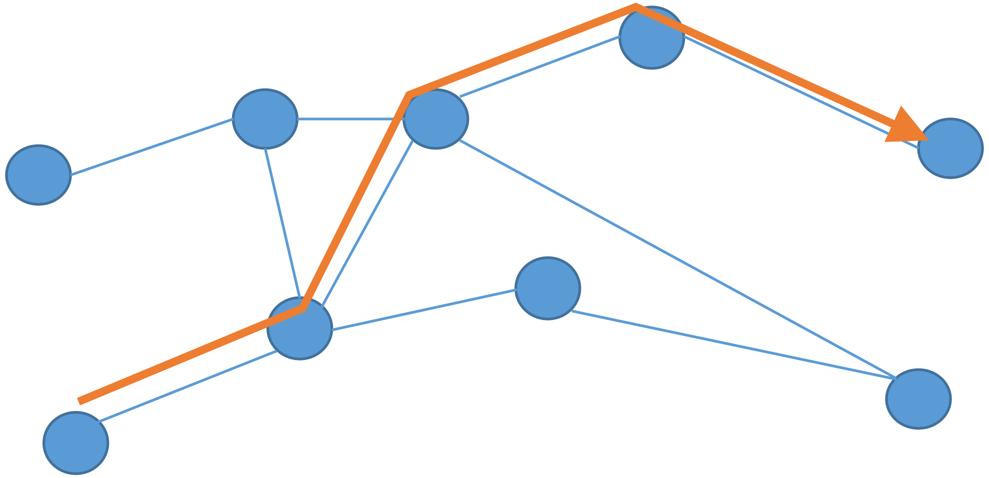
Architecture



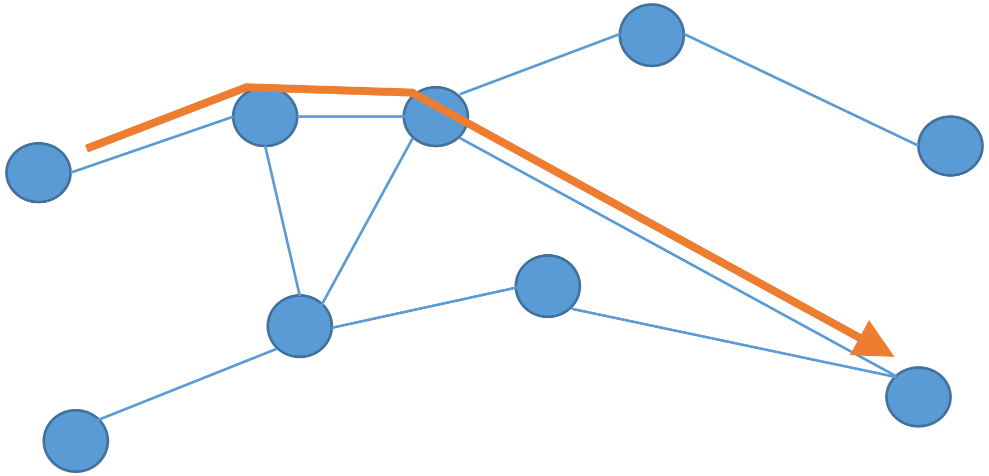
- For each Equivalence Class, calculates other candidate paths that preserve end-to-end characteristics.
- Candidates are "better" in terms of some metric (path length for instance).

Example: Shorter Path alternatives

IPv4 Equivalence Class: 3232235779 - 3232235779

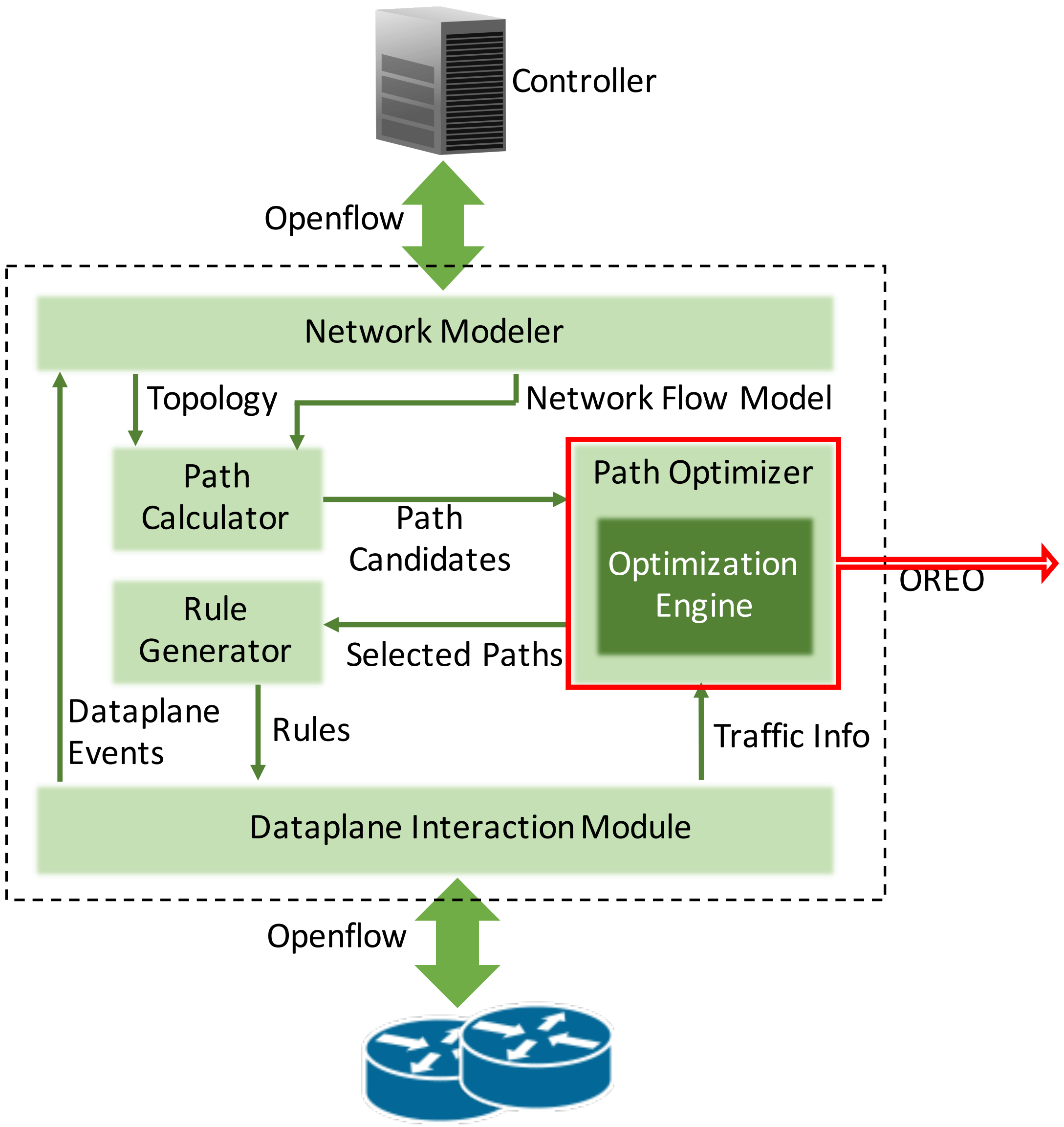


IPv4 Equivalence Class: 3232236035 - 3232236035



Other Equivalence Classes: Drop

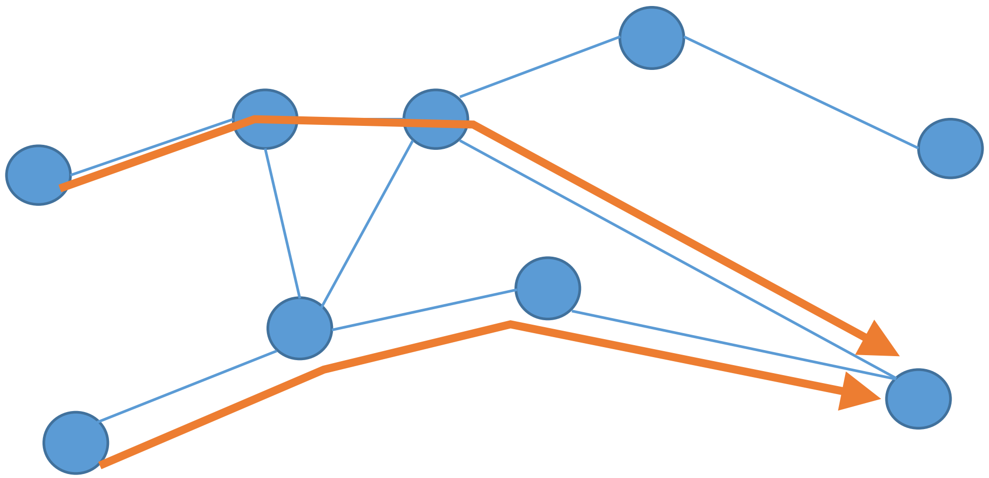
Architecture



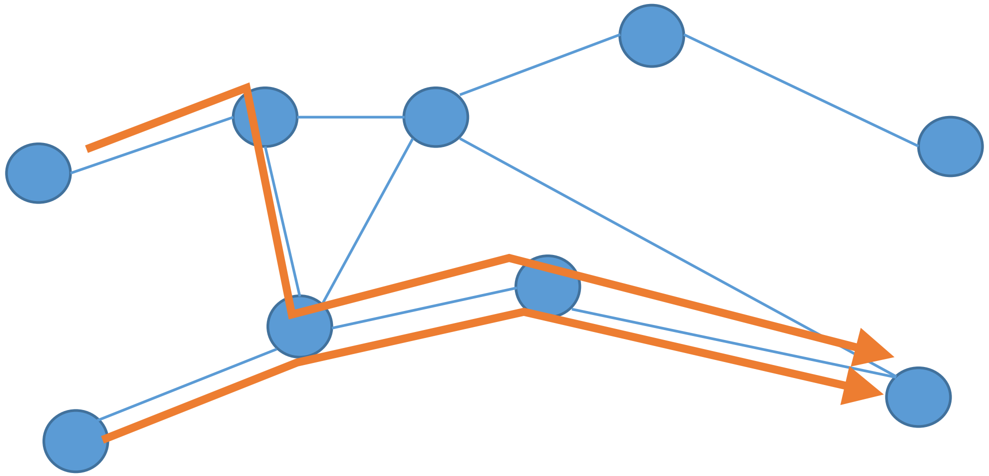
- Chooses the best combination of paths from among the candidates.
- Solves a multi-objective optimization problem.
- Currently experimenting with Linear Programming solvers.
- Weighted linear combination of objectives.

Example: Multiple Objectives

IPv4 Equivalence Class: 3232236035 - 3232236035



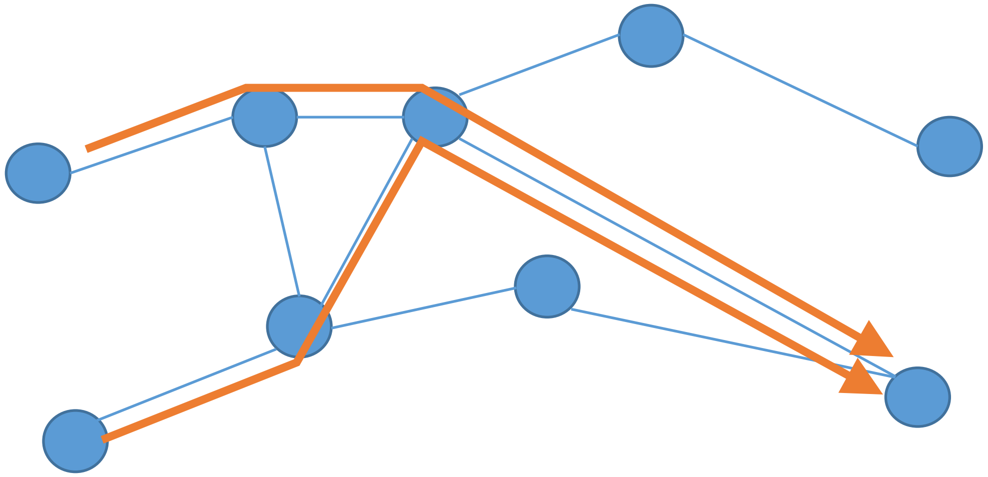
Combination that optimizes path length



Combination that optimizes switch memory use

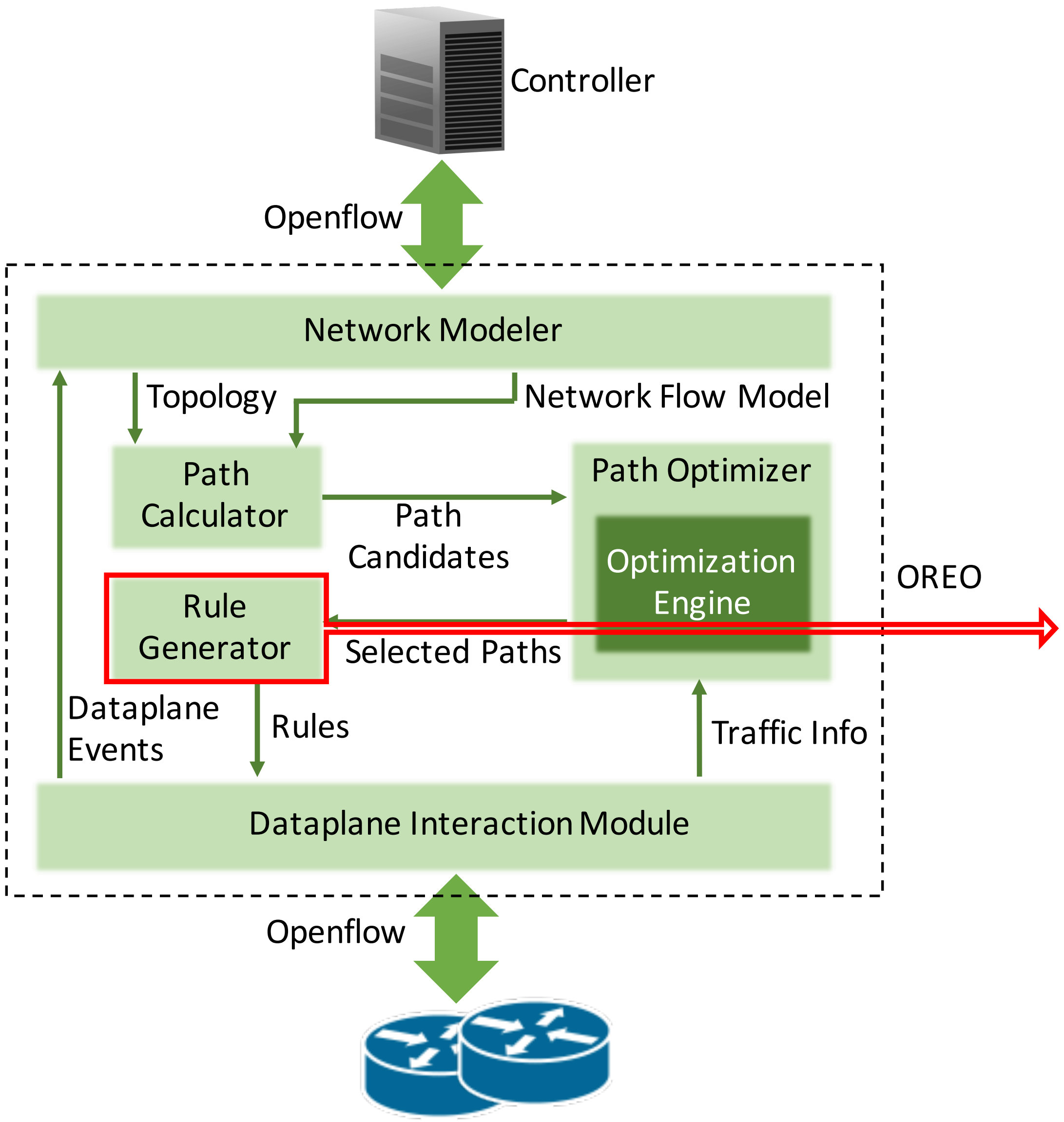
Example: Multiple Objectives

IPv4 Equivalence Class: 3232236035 - 3232236035



Combination that optimizes both goals

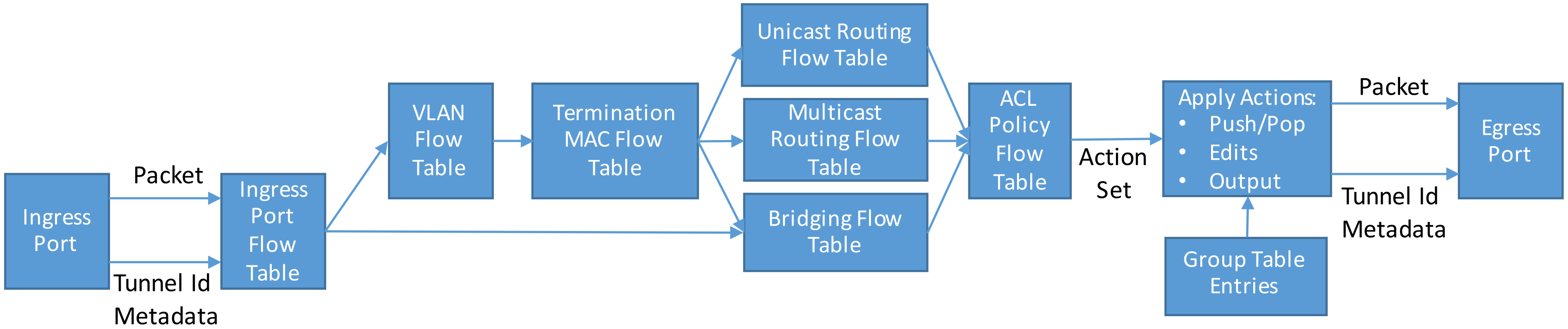
Architecture



- Maps each of the chosen paths into rules in the dataplane.
- Uses a switch pipeline abstraction similar to the flow table layouts in commercial forwarding devices.

Rule Generation: Switch Pipeline Abstraction

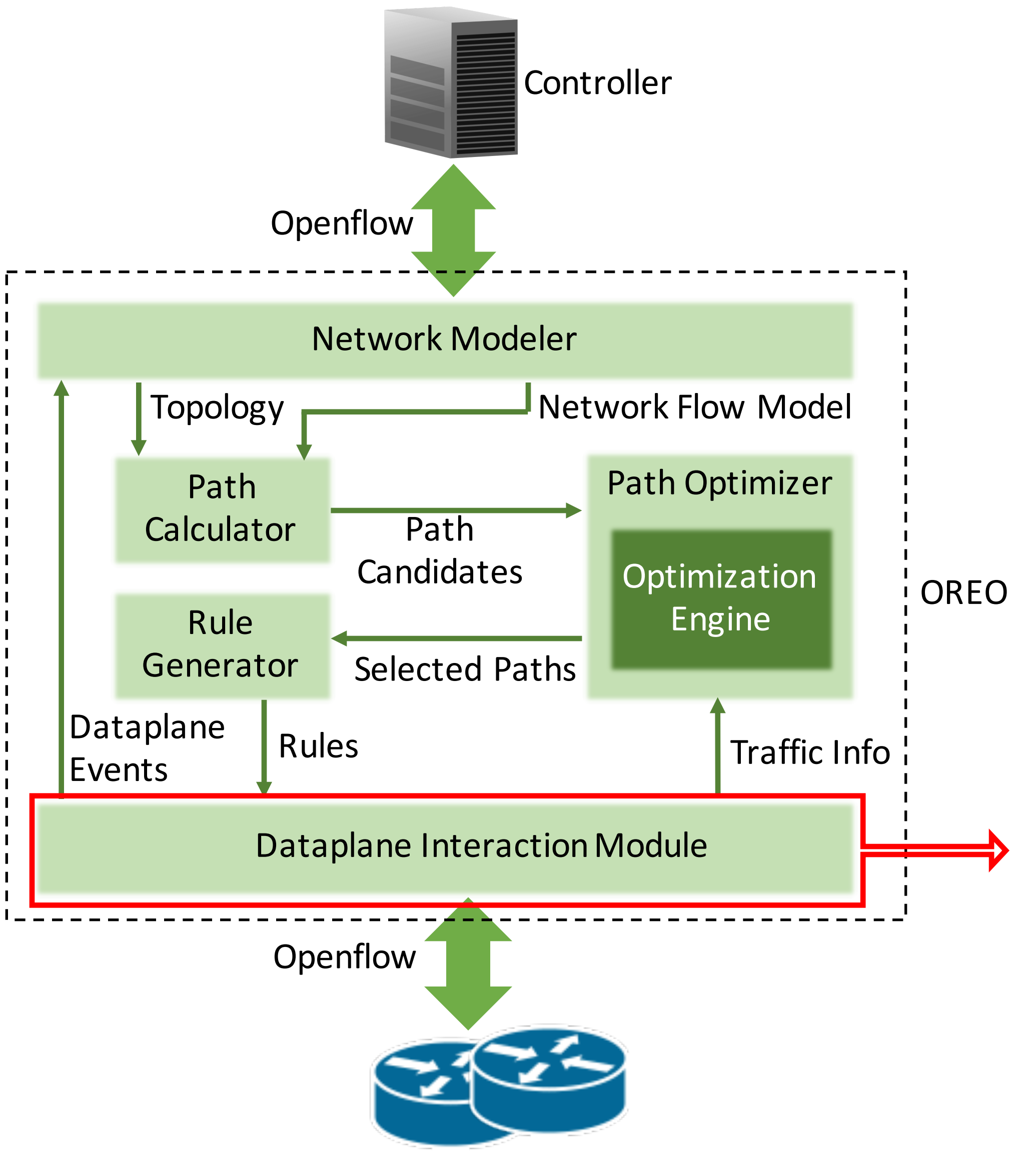
- Separate tables to perform each function.



Broadcom Switch Dataplane Abstraction

- If rules are generated after the optimization is finished, how does Oreo optimize rule memory consumption?
    - Approximate each equivalence class as a rule
    - Not always correct – Multiple ECs may be able to share the same rule
    - This approximation is known to work well

Architecture

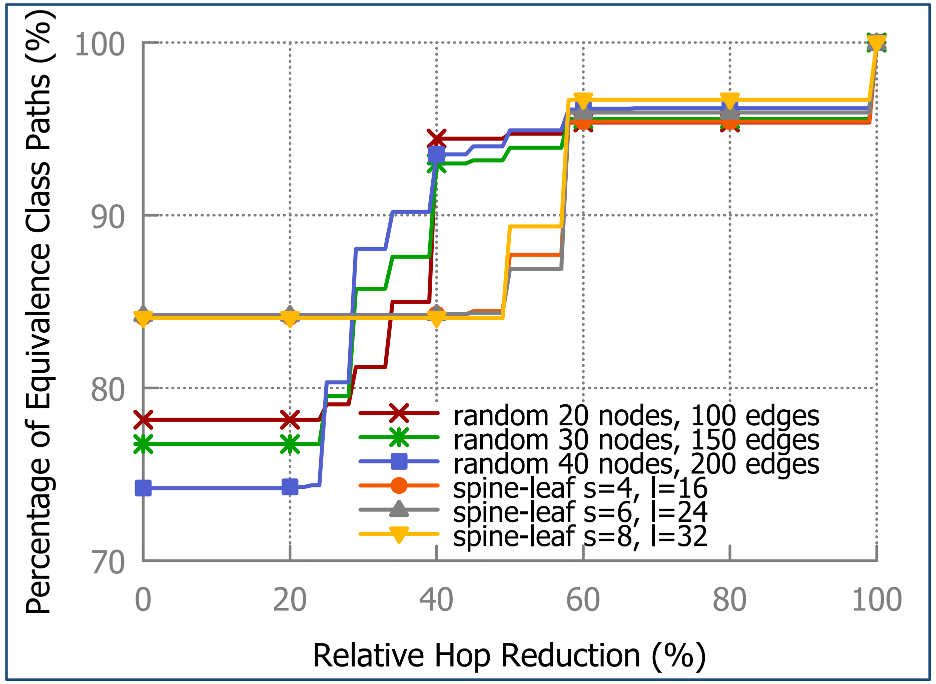


- Pushes out rules into the switches.
- Monitors and relays dataplane information (for dynamic optimization) to the higher layers.

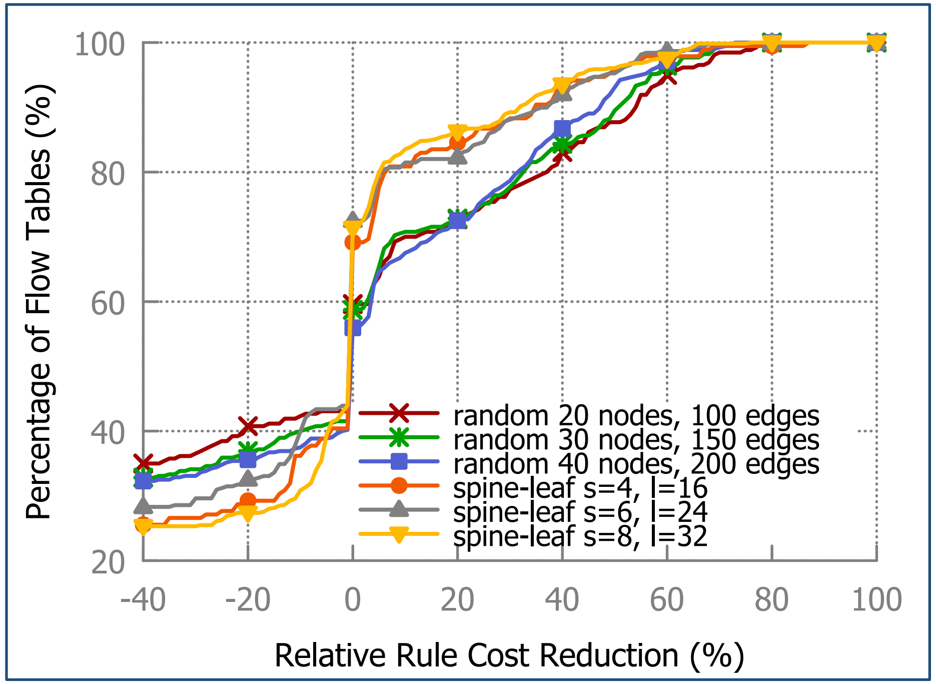
Initial Experiments

- Evaluates the linear-programming optimization mechanism.
- Networks with L3 forwarding and ACLs
- Uses equivalence classes and model defined by *Veriflow.*
- Optimizes a weighted sum of switch memory utilization and path length.
- K-Shortest path algorithm for path selection.
- Gurobi as the LP solver.

Experiment Results: Effectiveness

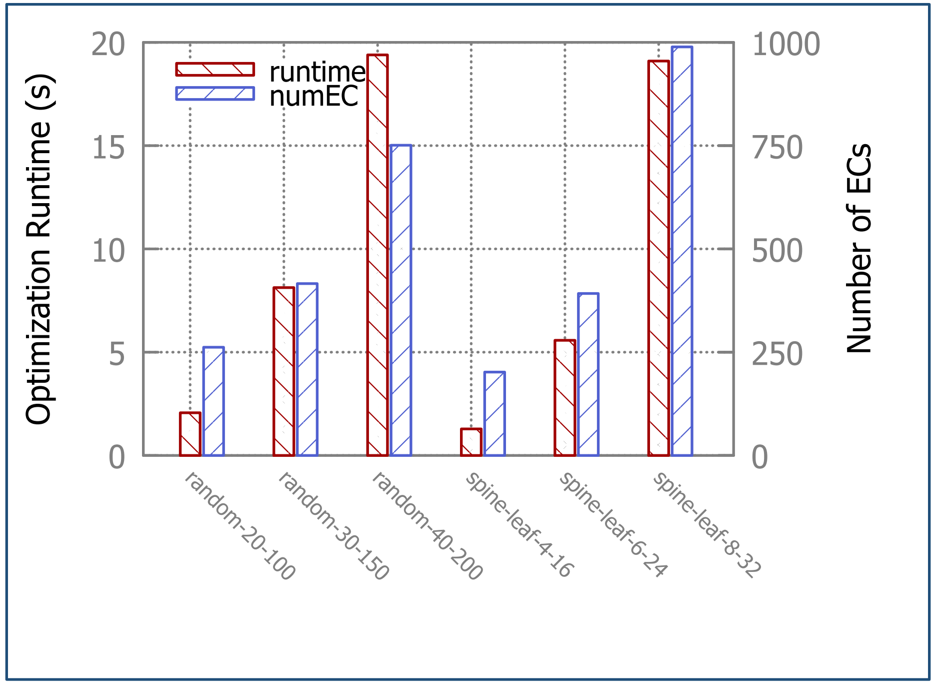


Reduction in path length

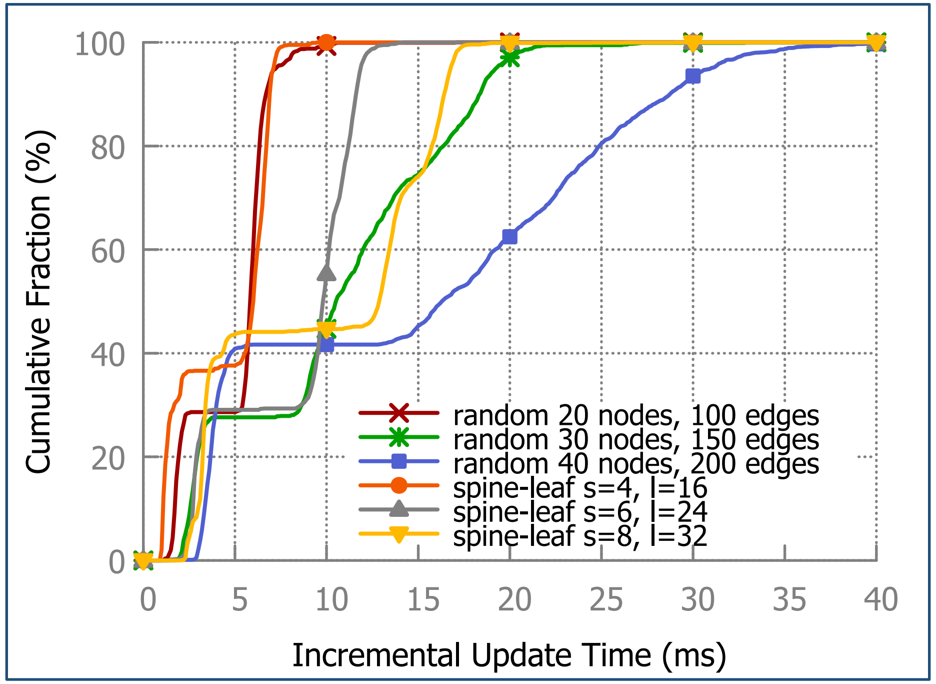


Reduction in switch memory consumption

Experiment Results: Scalability

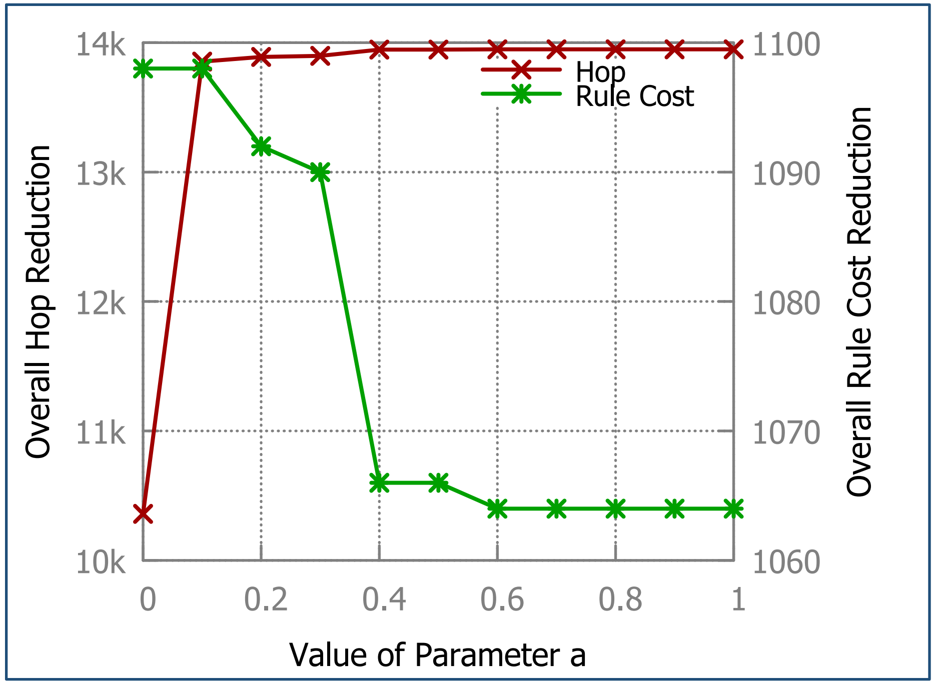


Time to perform optimization from scratch



Time to perform optimization on one extra EC

Experiment Results: Optimization Tradeoffs



Improvements for different weights (*a* is the weight assigned to Hop Count reduction)

Conclusion

- Network Security Policies are diverse and complicated.
- SDN hasn't helped with policy enforcement, due to the overhead of performance management.
- Oreo removes the performance burden from the administrator.
- Transparently optimizes the dataplane, while retaining end-to-end reachability characteristics.
- Network-wide, model-based optimization.
- Initial results indicate that the approach is effective, fast and scalable.

Questions?