Symphony: An Automated Approach to Detecting and Orchestrating Consistency Requirements

Saim Salman (saim_salman@brown.edu), Theophilus Benson (theophilus_benson@brown.edu)
Brown University

Problem

To build a redesigned SDN Control Plane that automatically infers the optimal consistency model and dynamically provides these models to different SDNApps.

Challenges

Capturing SDNApp Requirements to choose the appropriate consistency model:

- If the source code of the SDNApp is provided we can use **symbolic execution** to extract and infer SDNApp design.
- If the source code is absent we will design a **Domain Specific Language (DSL)** that enables the SDNApp developer to express predicates defining key behavior.

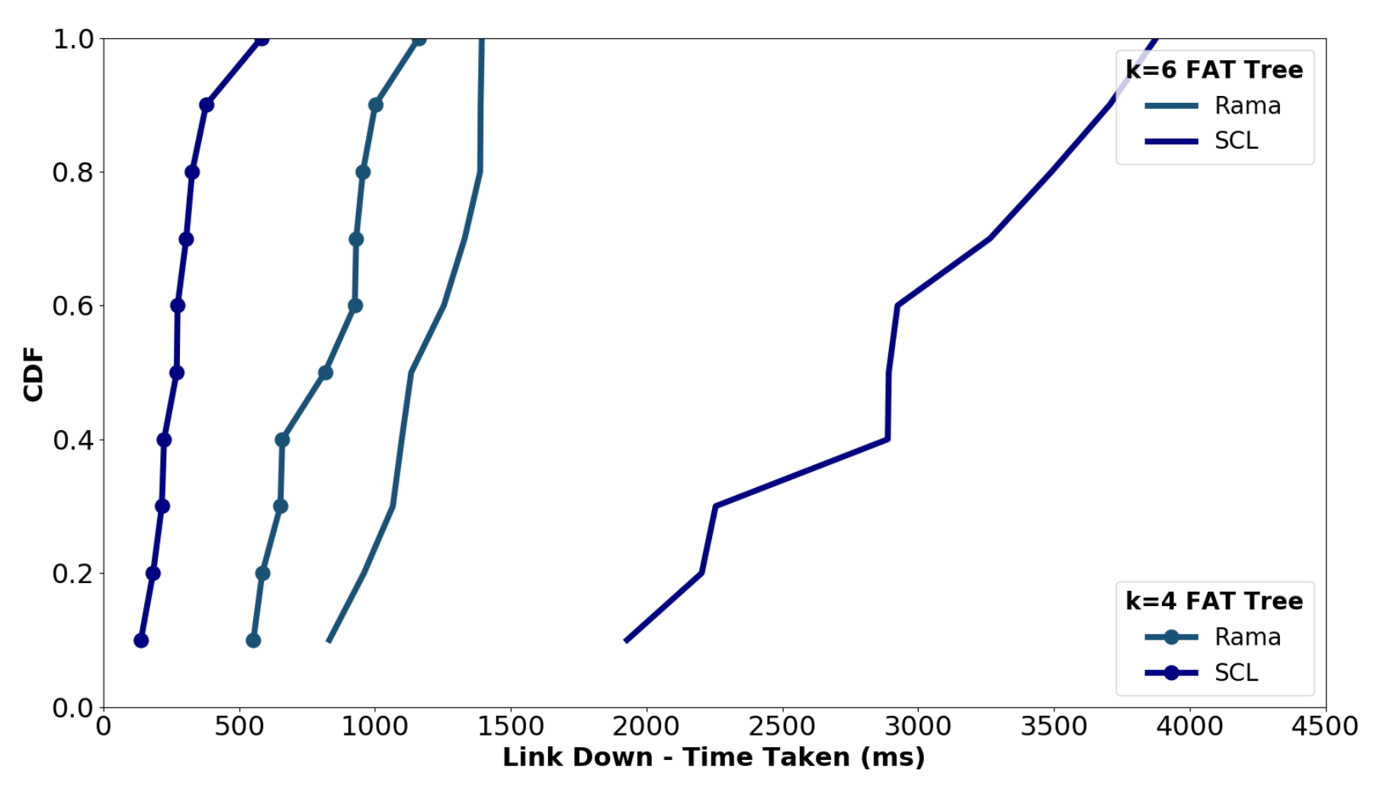
Effectively Supporting Concurrent Models:

- We will build on in-network flexibility by leveraging unique properties of programmable switches.
- We will also explore offloading consensus functionality to the switches to further improve speed and efficiency.
- At the controller side, we will explore efficient abstractions for presenting and enabling network wide transactions.

Motivation

SCL (Eventual)	Rama (Strong)
Deterministic	Deterministic
Proactive Applications	***Single Threaded***
Trigger Recomputation	
Idempotent Behavior	

Consistency models limits SDNApp design



No model provides strictly better performance

SDNApp	Consistency Model	
	Rama (Strong)	*SCL (Eventual)*
NAT	X	***X***
Routing	X	***X***
Load Balancer	X	
Stateful Firewall	X	

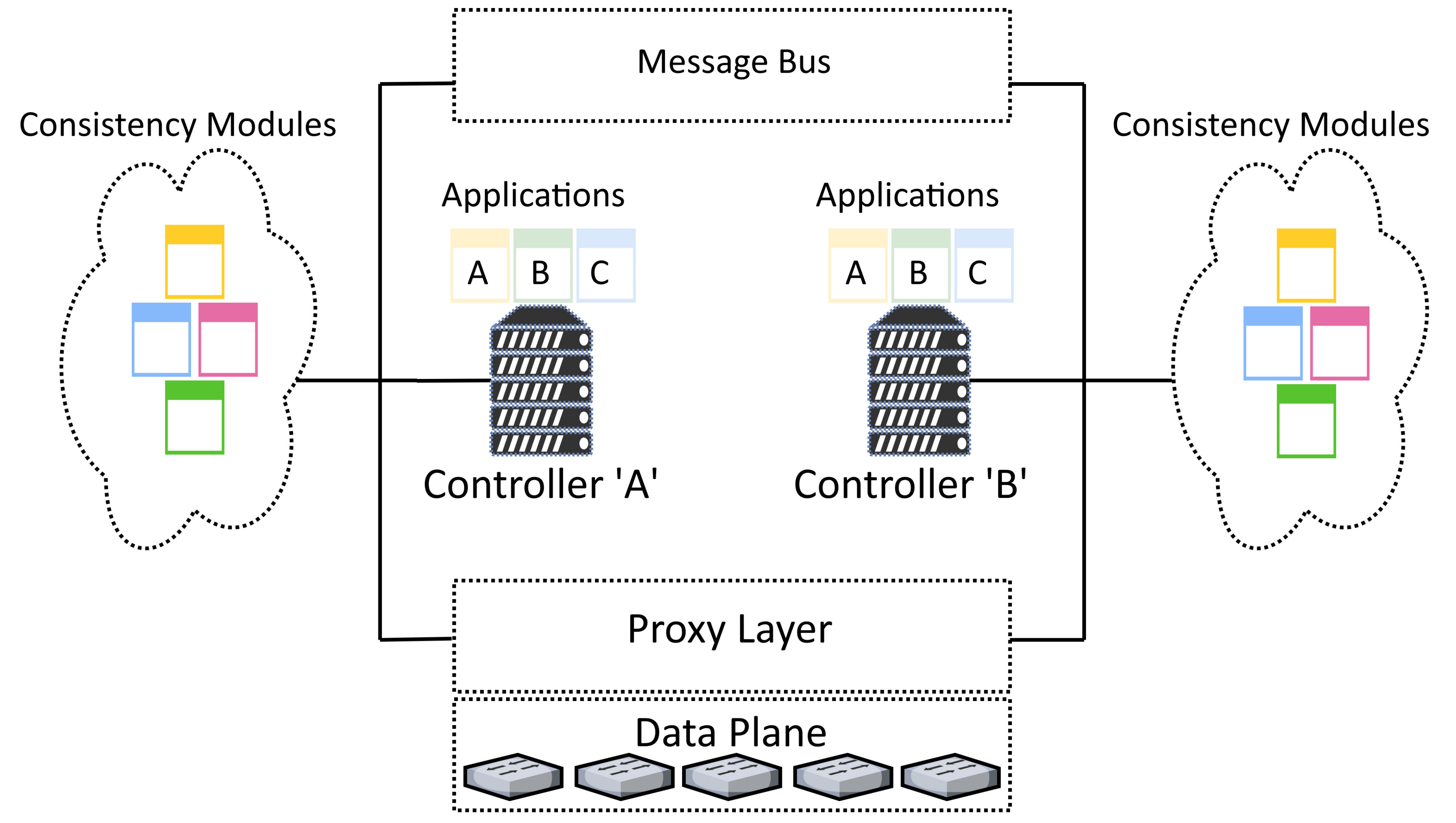
SDNApp Functionality limits applicable models

Takeaways:

- The SDNApps design determines which set of consistency models are suitable for the SDNApp – ***design limits choice***
- Under different demands, different consistency models may prove more efficient, thus the optimal model is dependent on the expected demand – ***demand determines optimal choice***

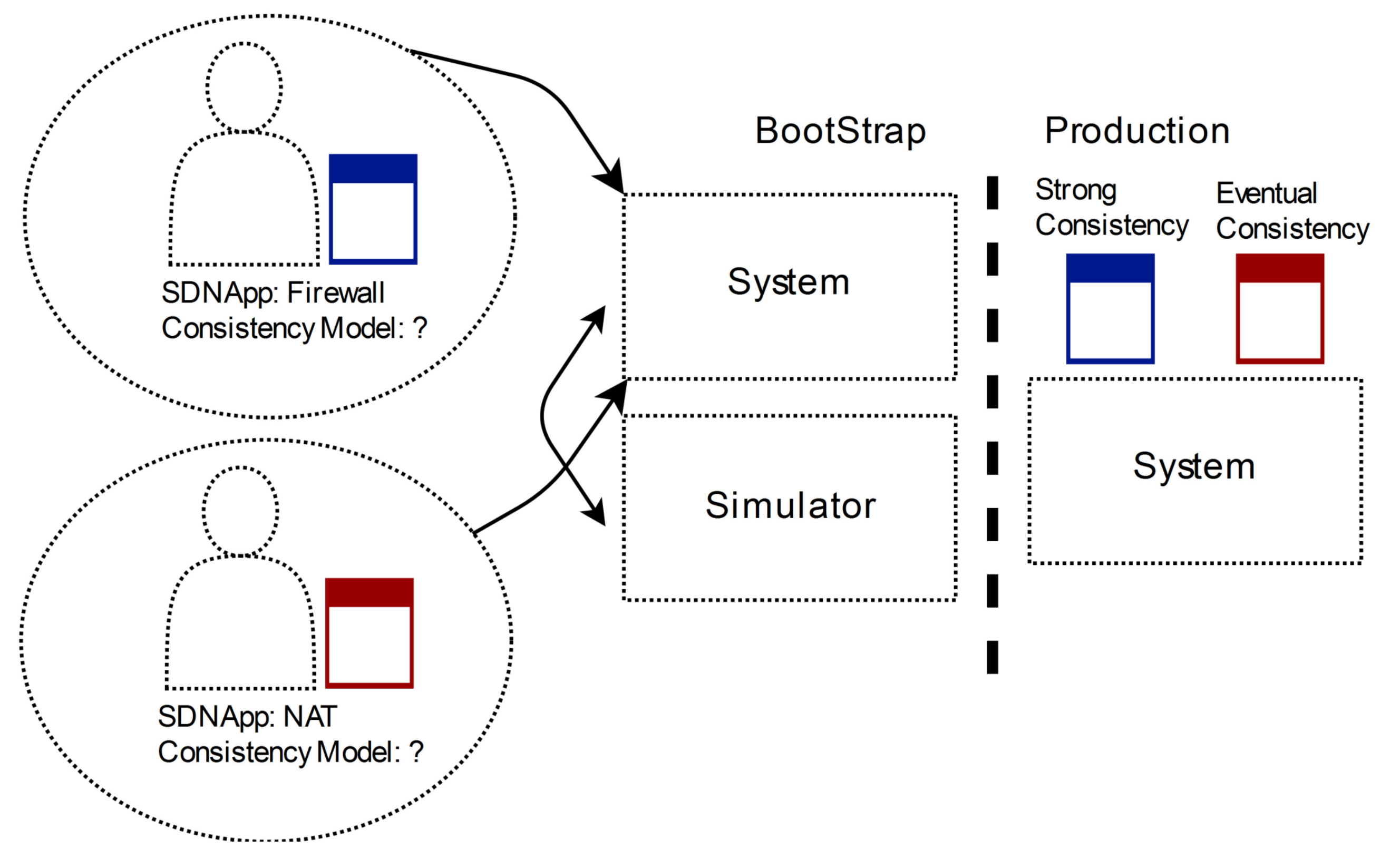
Design

- **Consistency Module:** Each consistency module would be associated with a specific consistency type and would provide a set of methods.
- **Message Bus:** Provides control traffic between different controller instances.
- **Proxy Layer:** Extra layer above each switch so consistency models can provide extra functionality at the switch for added performance.



Workflow

1. The developer **will describe consistency invariants** of a specific SDNApp through a simplistic **Domain Specific Language (DSL)** language.
2. Our system will **evaluate the SDNApp** across different models within our simulator to determine the set of **consistency models** that are appropriate for the SDNApp.
3. Given this list of appropriate models, the SDN App developer or network operator can **pick the model which optimizes their personal objectives**, e.g., performance and configure the SDN Controller to apply this model to the SDNApp.


SDNApp: Firewall
Consistency Model: ?
SDNApp: NAT
Consistency Model: ?
BootStrap
System
Simulator
Production
Strong Consistency
Eventual Consistency
System