

# NETWORK CONFIGURATION IS HARD!

- High-level tasks are realized through low-level commands and scripts: **hard to understand**
- Distributed configuration: **hard to manage**
- Variety of network-wide tasks cause changes to the network: **lots of dynamics**
- No changes are checked for correctness:  
**error-prone**

**20%** make changes more than once a day

**89%** are *never* completely certain that changes will not introduce a new bug

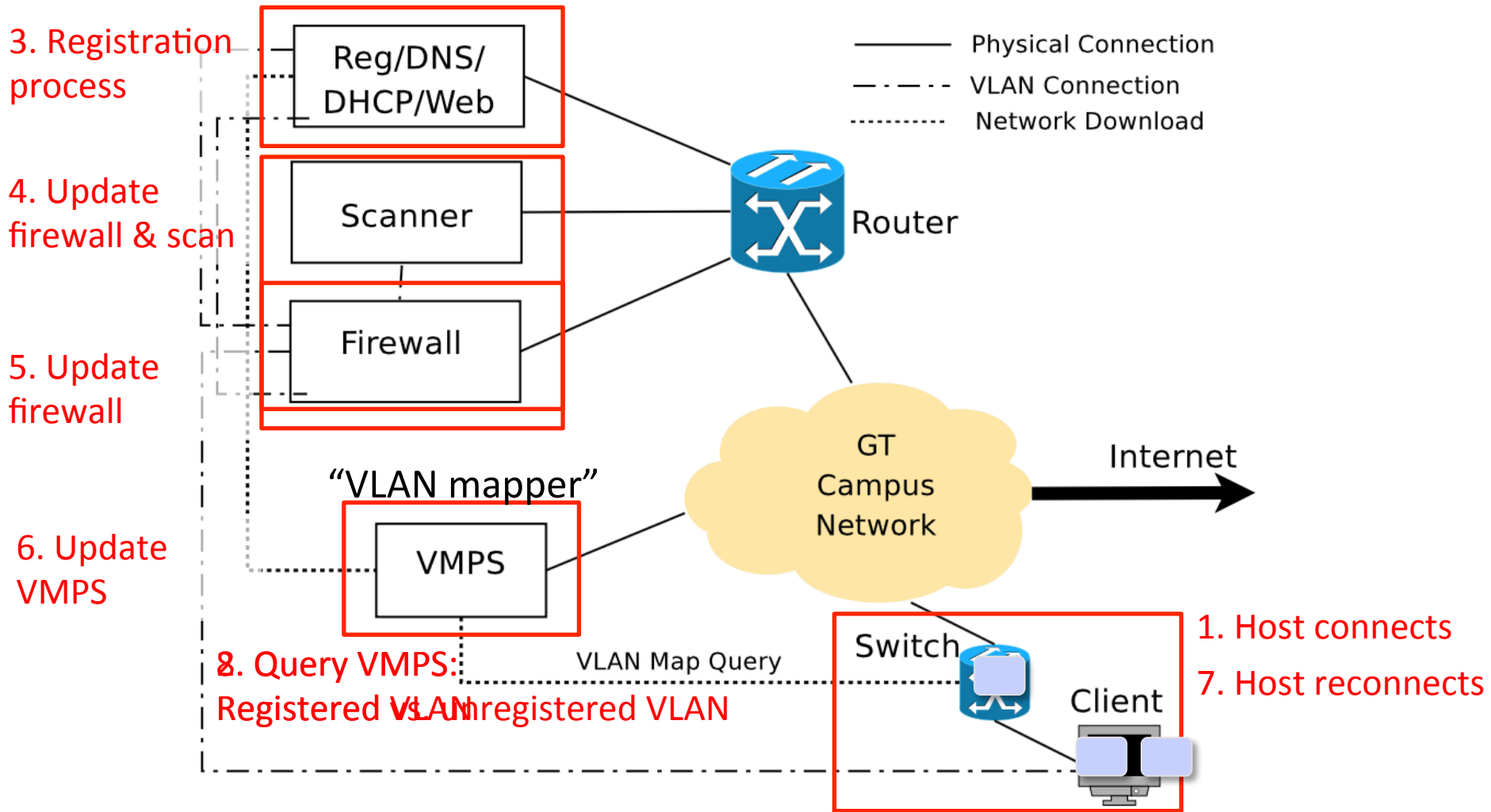
**82%** are concerned changes might break existing functionality unrelated to the changes

**20%** make changes more than once a day

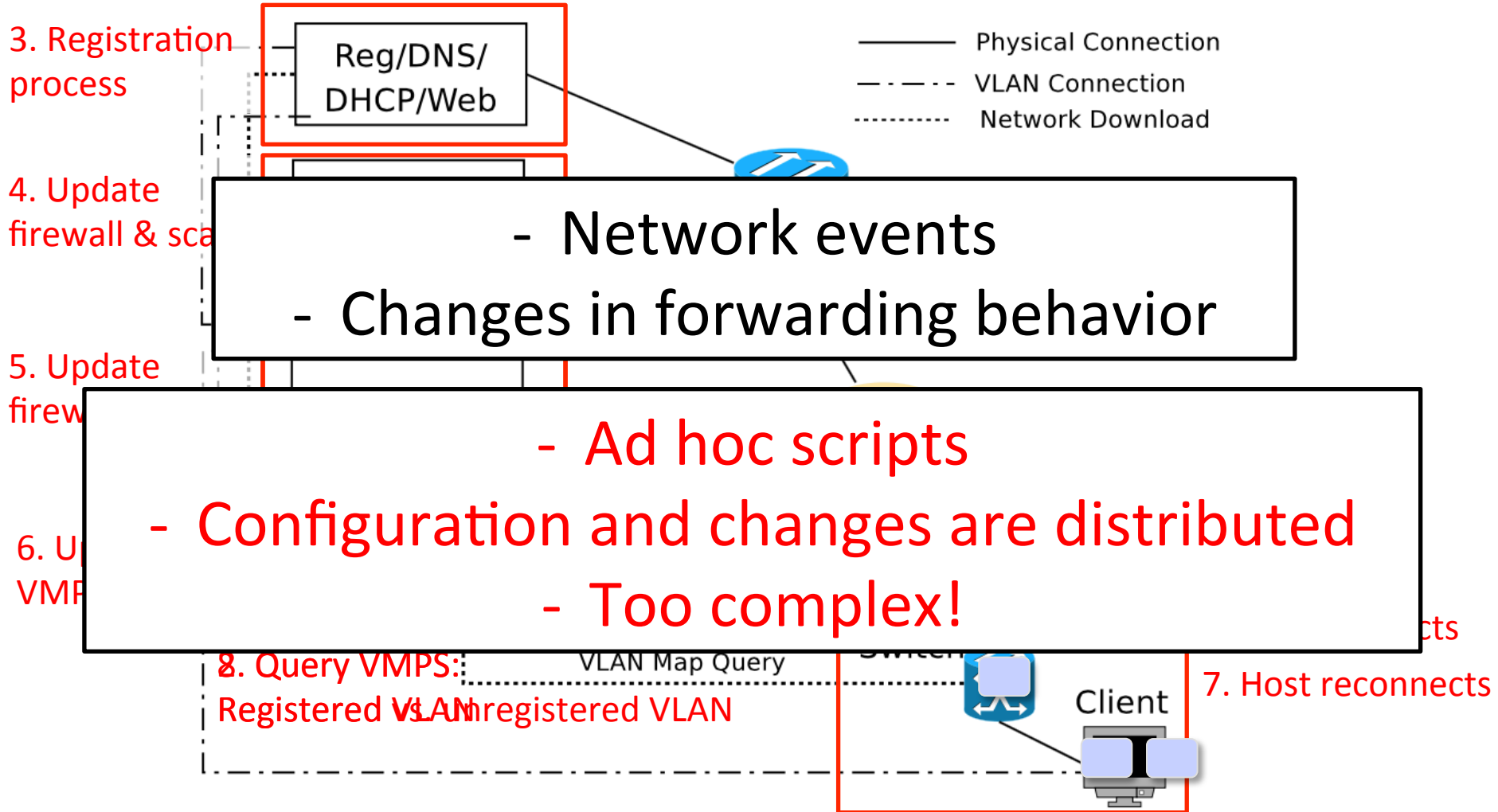
**89%** are *never* completely certain that changes will not introduce a new bug

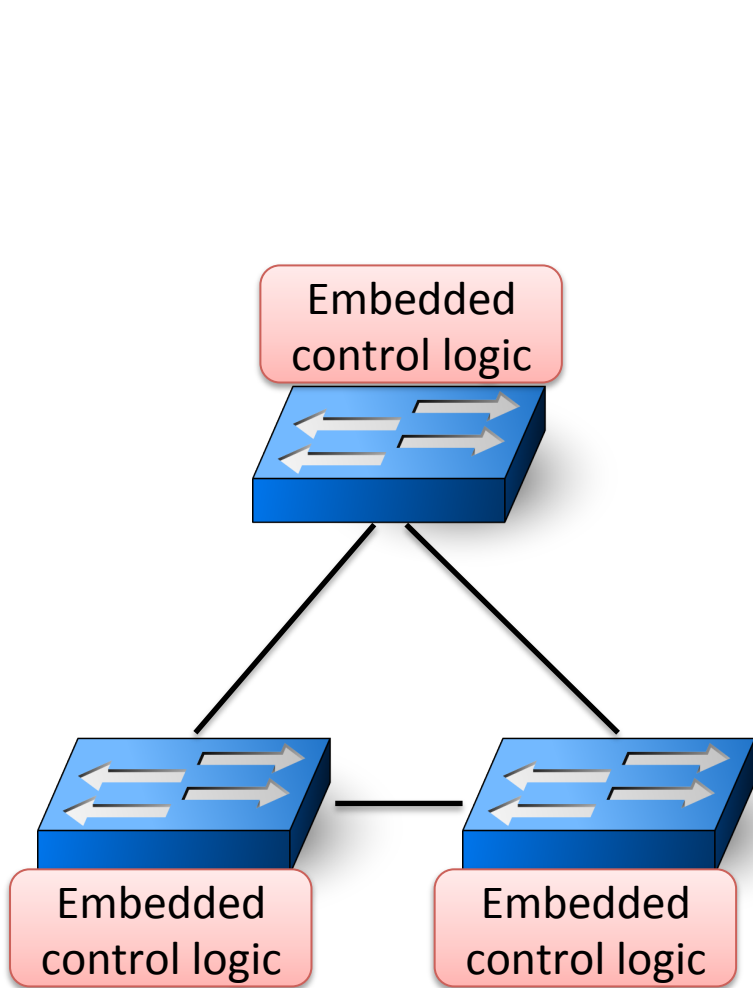
“You should track down those 10-20% of operators who say they are always certain. They are LYING.”

# MOTIVATING EXAMPLE: THE START SYSTEM

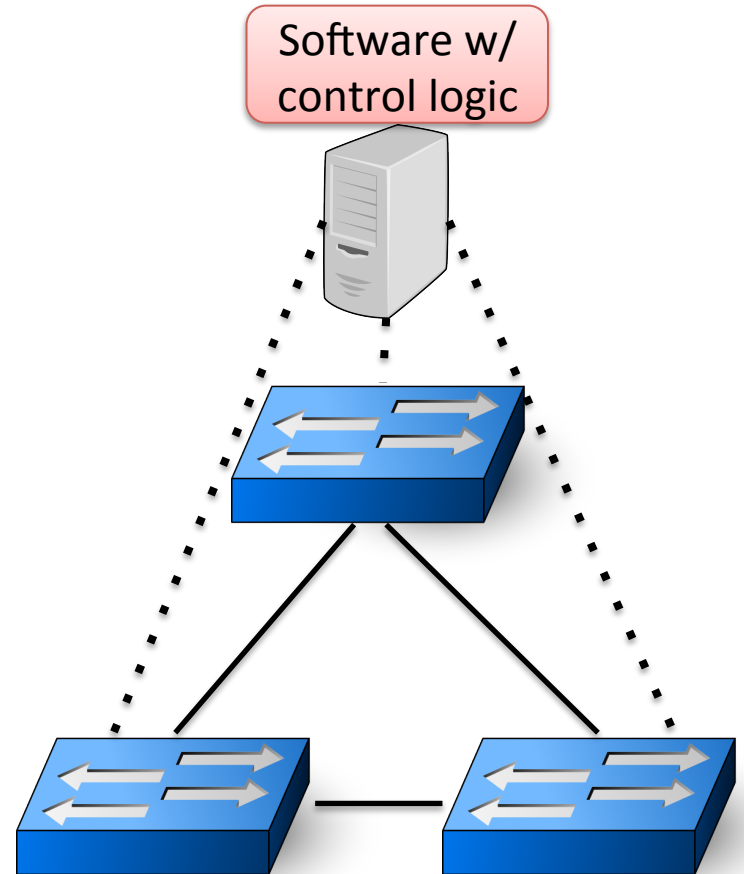


# MOTIVATING EXAMPLE: THE START SYSTEM





Traditional network



SDN

# SDN IS NOT A SILVER BULLET

- Low-level commands & scripts:  
**hard to understand**

Programs: e.g., C++,  
Java, Python, Pyretic

- Distributed configuration:  
**hard to manage**

Central control

- Many network-wide tasks, lots  
of changes: **lots of dynamics**

Unsolved

- No correctness guarantee:  
**error-prone**

Unsolved

- Guidance on how to implement a network control program
  - How to provide *dynamic control* that handles arbitrary network events
  - E.g, Intrusion detection, traffic load shift, etc
- Verification and guarantees of program's correctness
- Huge missed opportunities in software



- Network traffic
  - Traffic load increase/decrease, security incidents
- User-specific
  - User authentication, excessive data usage
- Data-plane events
  - Topology change, switch/link failures
- ...

# DIFFERENT REACTIONS TO AN EVENT

Event

Operators

Reaction

Host is infected!



“Only block that infected host”



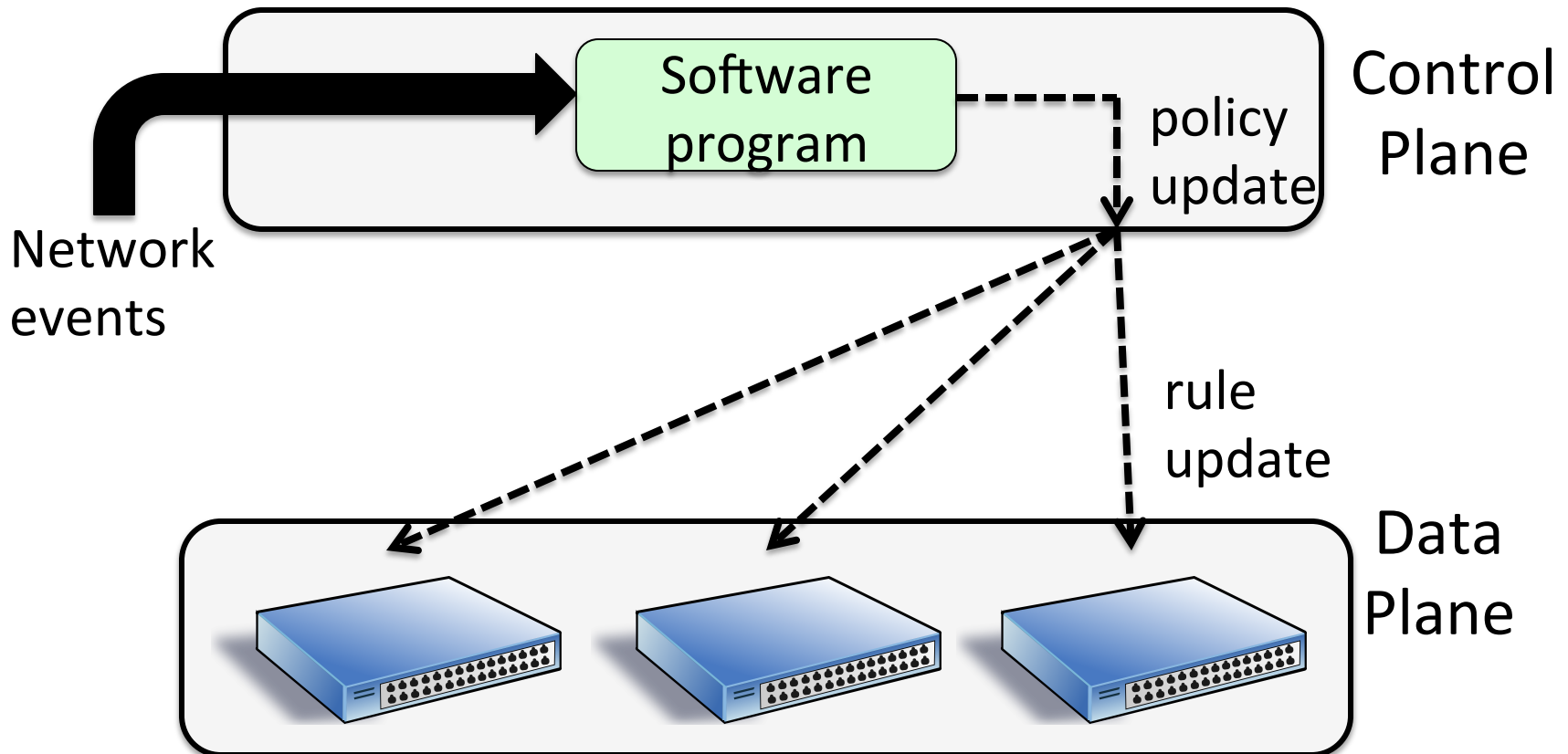
“Block all communications in the network!”



“Direct communication to our internal honeypot”

**Network events and dynamic reactions to them should be programmatically encoded in the network control program by operators**

- **Software program** that embeds event – reaction relationships



How to **embed event-reaction logic** in software?

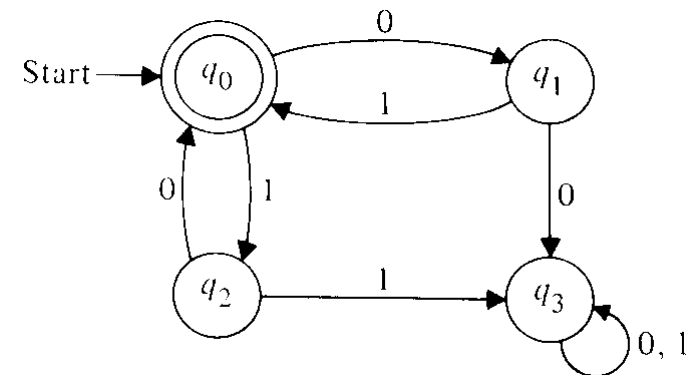
How to **verify** that the program will make  
**changes correctly?**

**Kinetic tackles these questions**

- **Domain specific** language and control platform
- Helps create SDN control programs that **embed custom event-reaction** relationships
- **Verifies** program's **correctness**

# OUR APPROACH

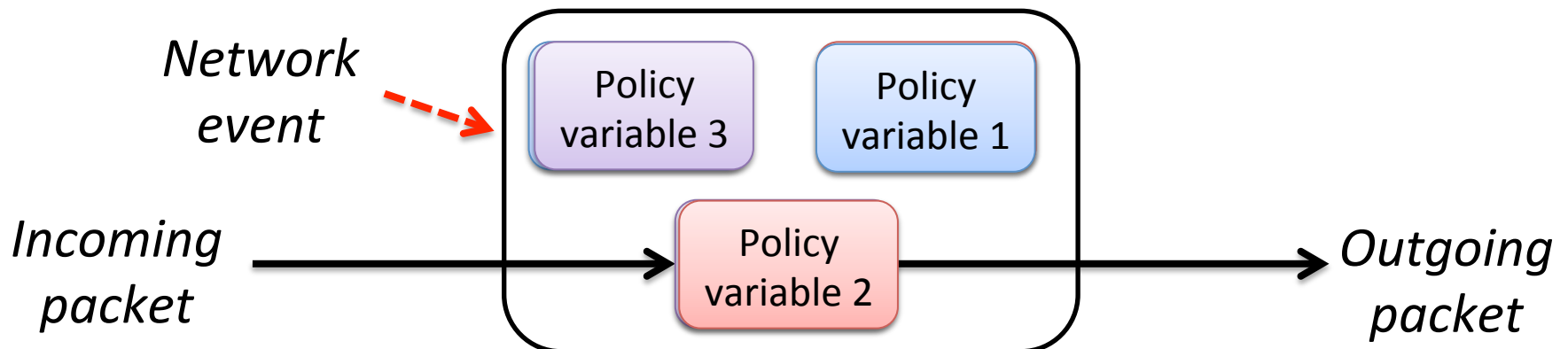
- Domain specific language
  - Constrained, but structured
- Express changing behavior as a *finite state machine*
- Verify program's correctness with a model checker (NuSMV)



- Embedded in Python
- Borrows some abstractions from Pyretic
  - Encodes forwarding behavior in a *policy variable*

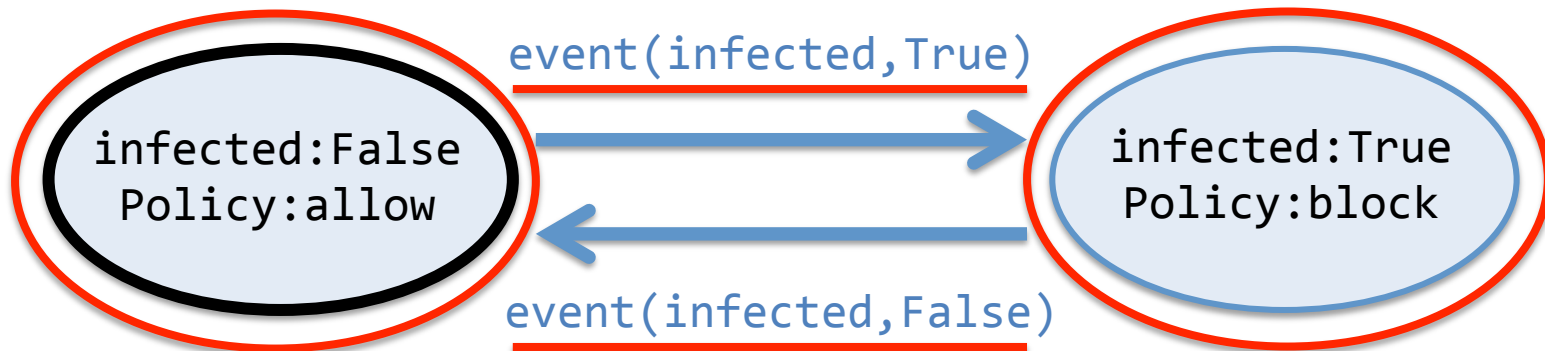


- New constructs and functions to express policies that respond to *changing conditions*



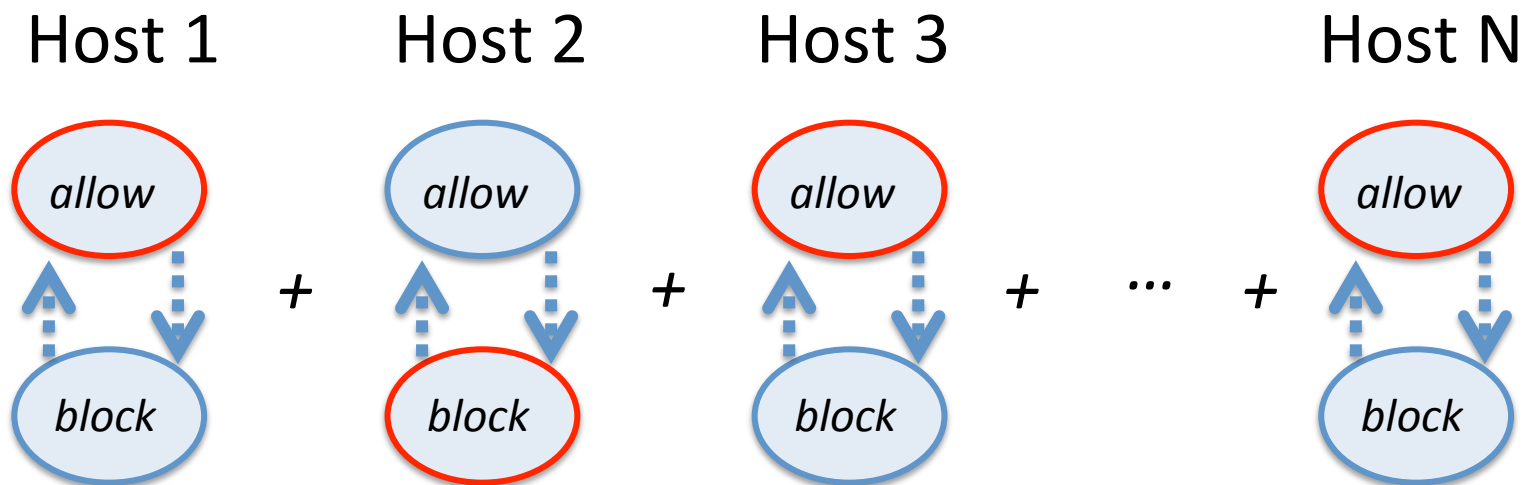


- **Event:** *infected*
- **State:** policy variable's value
  - *allow* or *block* packet



There are many different flows  
*Each flow can have its own independent FSM*

- FSM instance is instantiated per flow



# of hosts:  $N$

Total # of states:  $2N$

Total # of transitions:  $2N$

State representation is **Linear** in  $N$   
(instead of geometric)

- In IDS example, flow is defined by source IP address (host)
- Other policies may require more flexibility (e.g., need to group packets by location)
- **Located Packet Equivalence Class (LPEC)**
  - Programmer abstraction to define *flow*

```
def lpec(pkt):  
    return match(dstip=pkt['dstip'])
```

- Kinetic verifies correctness of the program
  - User-specified temporal properties
  - Verifies ***current and future*** forwarding behavior based on network events
- Verification process is ***automated***
  - Constrained but structured language allows automatic parsing and translation of program
- Verification runs ***before program's deployment***

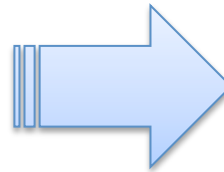
## Kinetic program

```
@transition
def infected(self):
    self.case(occured(self.event),self.event)

@transition
def policy(self):
    self.case(is_true(V('infected')),C(drop))
    self.default(C(identity))

self.fsm_def = FSMDef(
    infected=FSMVar(type=BoolType(),
                    init=False,
                    trans=infected),
    policy=FSMVar(type=Type(Policy,{drop,
                              identity}),
                  init=identity,
                  trans=policy))
```

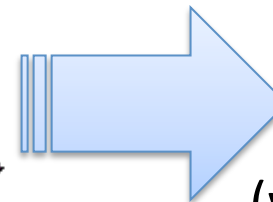
Automatically  
generates



## NuSMV FSM model

```
MODULE main
VAR
    policy    : {identity,drop};
    infected  : boolean;
ASSIGN
    init(policy) := identity;
    init(infected) := FALSE;
    next(policy) :=
        case
            infected : drop;
            TRUE     : identity;
        esac;
    next(infected) :=
        case
            TRUE     : {FALSE,TRUE};
        esac;
```

User-specified  
temporal properties



**True or False**  
(w/ counter-example)

- If a host is infected, drop packets from that host

$\boxed{AG}(\text{infected} \rightarrow \boxed{AX}\text{policy}=\text{drop})$

For all possible transitions from  
current state,

For all current and future  
states,

For all possible transitions  
from current state,

For the next state,

- If host is authenticated either by Web or 802.1X, and is not infected, packets should never be dropped.

$AG ( (\text{authenticated\_web} | \text{authenticated\_1x}) \& !\text{infected} \rightarrow AX \text{policy} \neq \text{drop} )$

- Usability evaluation
  - User study against over 870 participants
  - Lines of code comparison with other SDN solutions
- Performance and scalability
  - Event handling and policy recompilation

- Demographic

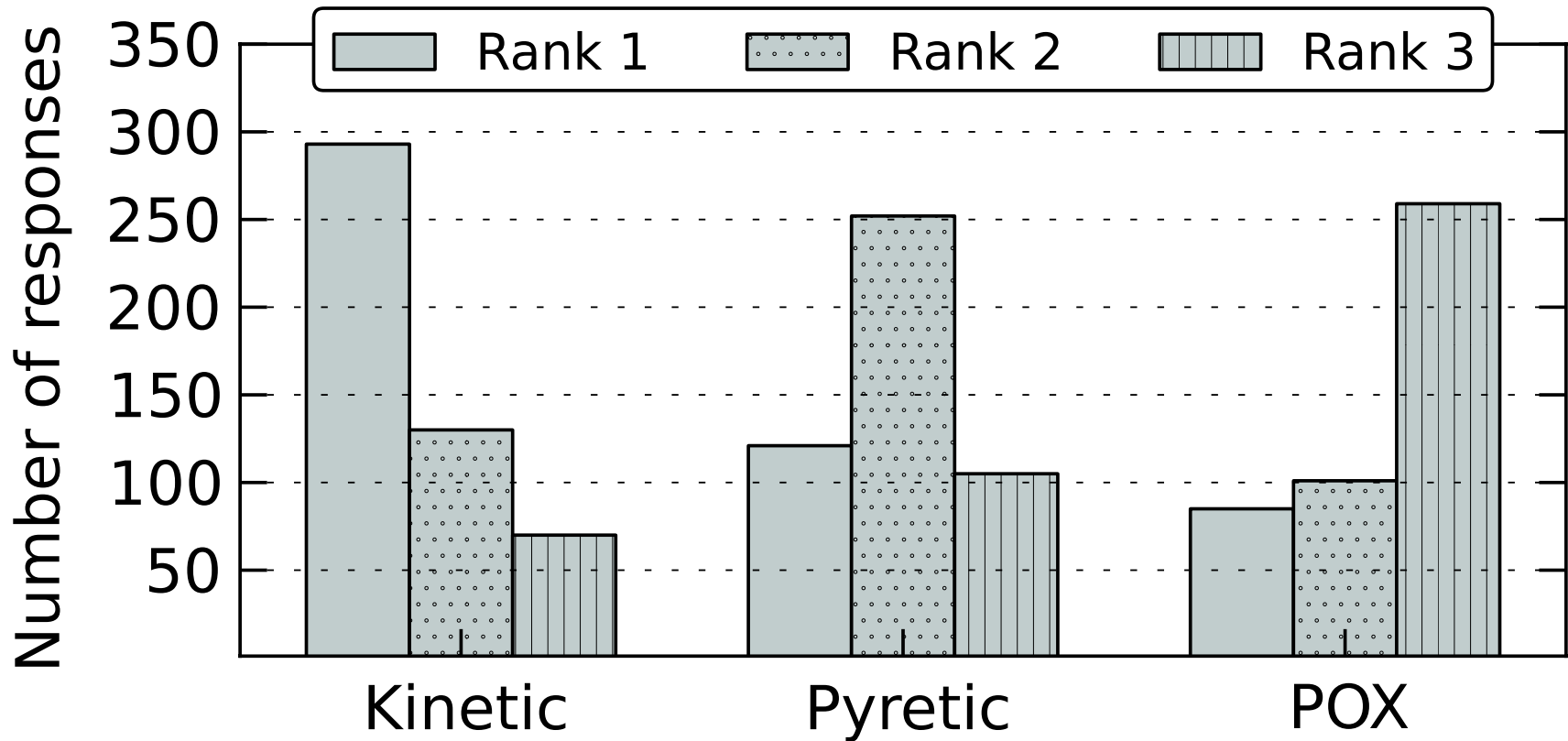
Profession		Experience (years)	
Operator	216	1	32
Developer	251	1-5	310
Student	123	5-10	187
Vendor	80	10-15	150
Manager	69	15-20	122
Other	138	> 20	73
<b>Total</b>	<b>877</b>		<b>874</b>

- Task

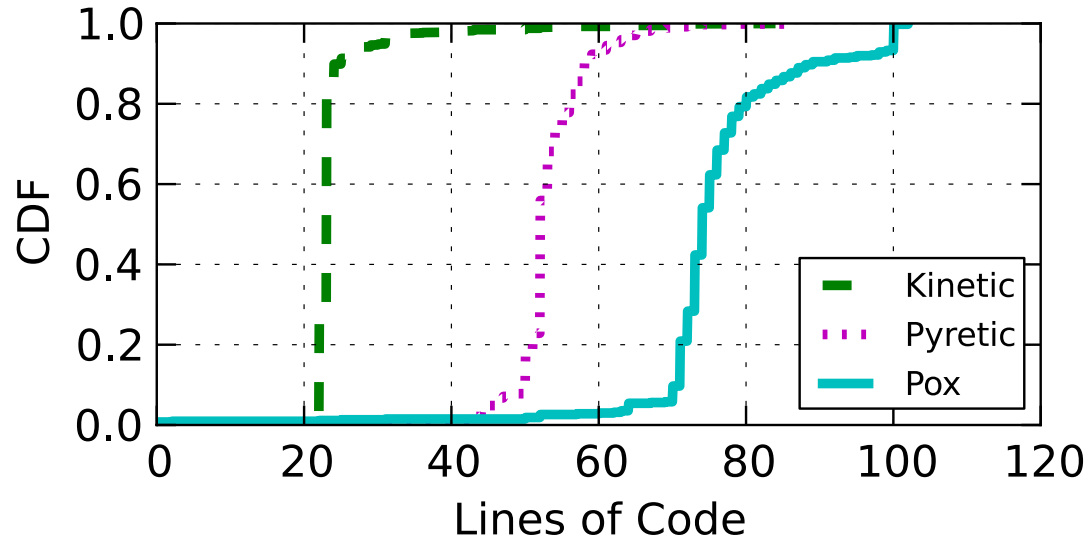
- Implement an enhanced IDS program with Kinetic, Pyretic, and POX.



# RANK PLATFORMS BY PREFERENCE



# LINES OF CODE COMPARISON



Programs	FL	POX	Pyretic	Kinetic
IDS/firewall	416	22	46	17
Mac learner	314	73	17	33
Server load balance	951	145	34	37
Stateful firewall	<i>None found</i>	<i>None found</i>	25	41
Usage-based rate limiter	<i>None found</i>	<i>None found</i>	<i>None found</i>	30

- Why did you like Kinetic?
  - **FSM-based structure and support for intuition**

*“Kinetic is more intuitive: the only things I need to do is to define the FSM variable”*

*“intuitive and easy to understand”*

*“Programming state transitions in FSMs makes much more sense”*

- **More concise**

*“reduces the number of lines of code”*

*“the logic is more concise”*

- Why *didn't* you like Kinetic?

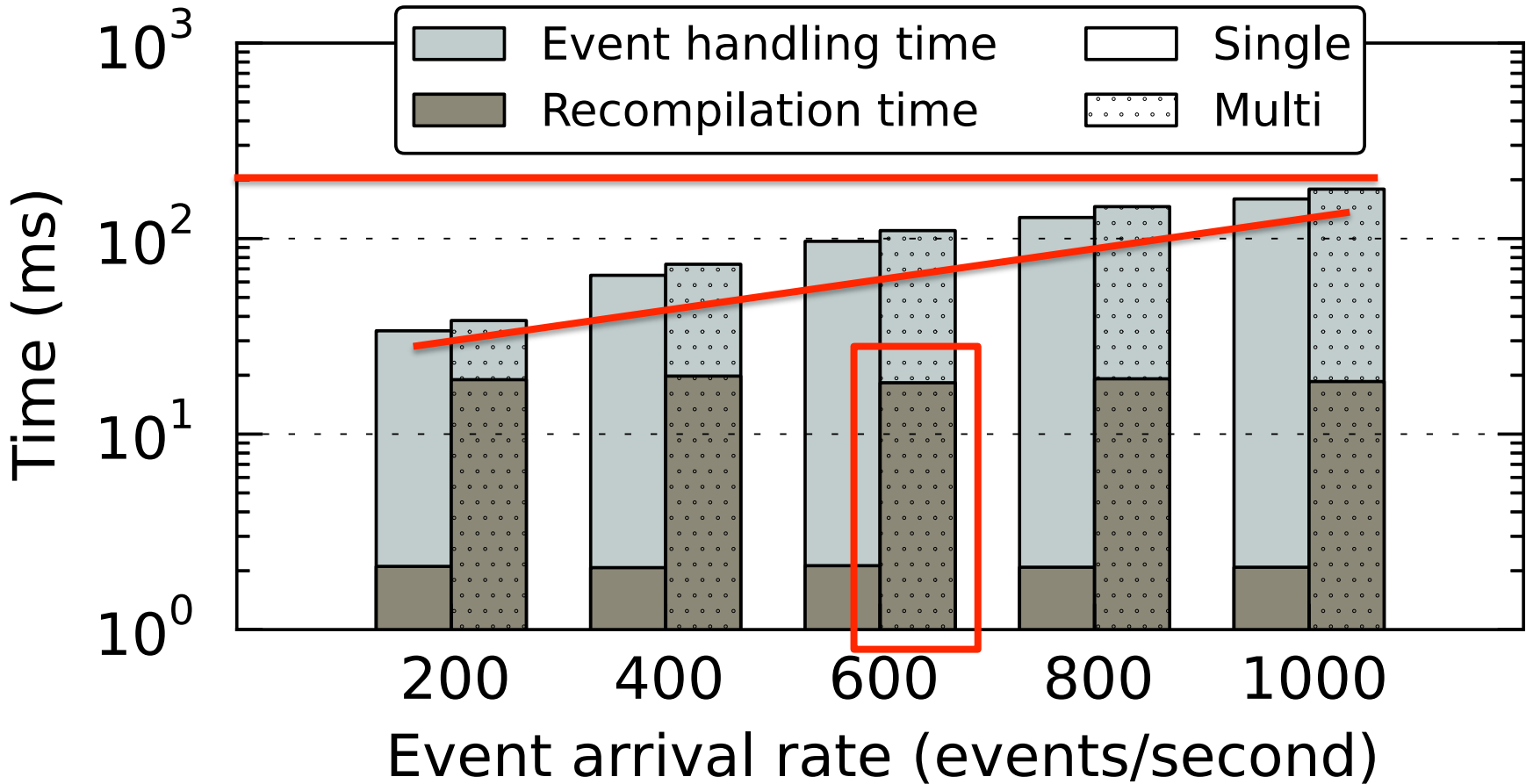
- **Steep learning curve**

- “Kinetic took less time and was actually more understandable ...[but] the structure was very cryptic”*

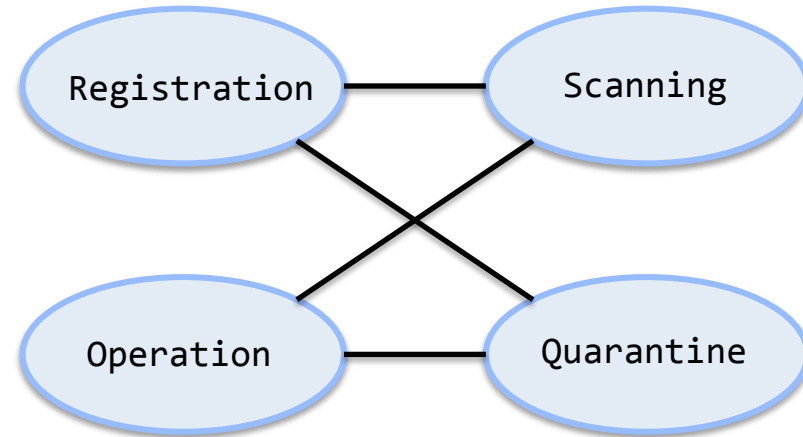
- **Not friendly when finding why program is wrong**

- “I spent a lot more time chasing down weird bugs I had because of things I left out or perhaps didn't understand”*

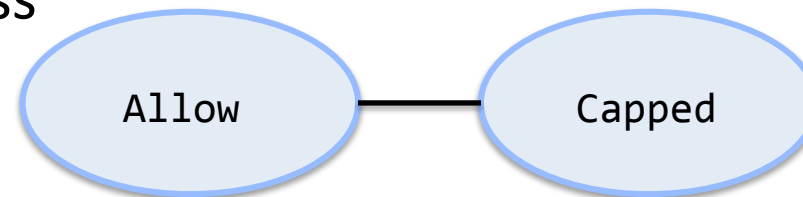
# Event handling and policy recompilation



- Campus network
  - Functional access control system
  - Deployed SDN-enabled switches over 3 buildings



- Home network
  - Usage-based access control
  - Deployed 21 SDN-enabled wireless routers over 3 continents
  - Jul., 2012 – Feb., 2014
  - Presented in ACM CHI 2015



- **Domain specific** language and control platform
  - Program **encodes event-reaction logic**
- **Extensive user study** shows that
  - Much **easier to express dynamics** in the network
  - Helps to **reduce lines of code**
- **Scales well** to large networks and lots of events
- **Verification process reduces bugs** in programs

- Combining with verifications in other stacks
  - Consistent updates to data plane
  - Verification of data-plane state
  
- More dynamic network policies
  - Should collect more real network policies
  - Need public repository



# THANK YOU

More about Kinetic:

<http://kinetic.noise.gatech.edu>

Contact:

[joonk@gatech.edu](mailto:joonk@gatech.edu)

## Questions?