• 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
SURVEY WITH NETWORK OPERATORS

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

1. Host connects
2. Query VMPS: Registered VLAN
3. Registration process
4. Update firewall & scan
5. Update firewall
6. Update VMPS
7. Host reconnects
8. Query VMPS: Registered VLAN

- Reg/DNS/DHCP/Web
- Scanner
- Firewall
- VMPS
- "VLAN mapper"

- GT Campus Network
- Internet
- Physical Connection
- VLAN Connection
- Network Download

Client

Switch

VMPS

"VLAN mapper"
MOTIVATING EXAMPLE: THE START SYSTEM

1. Host connects
2. Query VMPS: Registered vs. unregistered VLAN
3. Registration process
4. Update firewall & scan
5. Update firewall
6. Update VMPS
7. Host reconnects
8. Query VMPS: Registered VLAN

- Network events
- Changes in forwarding behavior
- Ad hoc scripts
- Configuration and changes are distributed
- Too complex!
SOFTWARE-DEFINED NETWORKING

Traditional network

SDN
SDN IS NOT A SILVER BULLET

- Low-level commands & scripts: hard to understand
- Distributed configuration: hard to manage
- Many network-wide tasks, lots of changes: lots of dynamics
- No correctness guarantee: error-prone

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

Central control

Unsolved
WHAT SDN PLATFORM NEEDS

• 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
DIFFERENT TYPES OF NETWORK EVENTS

• 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

<table>
<thead>
<tr>
<th>Event</th>
<th>Operators</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host is infected!</td>
<td></td>
<td>“Only block that infected host”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Block all communications in the network!”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Direct communication to our internal honeypot”</td>
</tr>
</tbody>
</table>
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

![Diagram of network control program](image)

- **Network events**
- **Software program**
- **Policy update**
- **Rule update**
- **Data Plane**
- **Control Plane**
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 \textit{policy variable}

\textit{Incoming packet} \quad \textit{Policy variable} \quad \textit{Outgoing packet}

• New constructs and functions to express policies that respond to \textit{changing conditions}

\textit{Incoming packet} \quad \textit{Network event} \quad \textit{Outgoing packet}
• 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

Host 1  Host 2  Host 3  Host N
allow  allow  allow  allow
|         |         |         |         |
| block   | block   | block   | block   |

# 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

```python
def lpec(pkt):
    return match(dstip=pkt['dstip'])
```
KINETIC VERIFICATION PROCESS

• 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))

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;

Automaticall generates

User-specified temporal properties

NuSMV Model Checker

True or False (w/ counter-example)
• If a host is infected, drop packets from that host

\[ \text{AG} \left( \text{infected} \rightarrow \text{AX policy}=\text{drop} \right) \]

For all possible transitions from current state,

For all current and future states,

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

\[ \text{AG} \left( \left( \text{authenticated_web} \mid \text{authenticated_1x} \right) \& \!\text{infected} \rightarrow \text{AX policy}!=\text{drop} \right) \]
EVALUATION

• Usability evaluation
  – User study against over 870 participants
  – Lines of code comparison with other SDN solutions

• Performance and scalability
  – Event handling and policy recompilation
KINETIC: USER STUDY

• Demographic

<table>
<thead>
<tr>
<th>Profession</th>
<th>Experience (years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>216</td>
</tr>
<tr>
<td>Developer</td>
<td>251</td>
</tr>
<tr>
<td>Student</td>
<td>123</td>
</tr>
<tr>
<td>Vendor</td>
<td>80</td>
</tr>
<tr>
<td>Manager</td>
<td>69</td>
</tr>
<tr>
<td>Other</td>
<td>138</td>
</tr>
<tr>
<td>Total</td>
<td>877</td>
</tr>
</tbody>
</table>

• Task

– Implement an enhanced IDS program with Kinetic, Pyretic, and POX.
RANK PLATFORMS BY PREFERENCE

Number of responses

<table>
<thead>
<tr>
<th>Platform</th>
<th>Rank 1</th>
<th>Rank 2</th>
<th>Rank 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinetic</td>
<td>350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pyretic</td>
<td></td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>POX</td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>Programs</td>
<td>FL</td>
<td>POX</td>
<td>Pyretic</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>IDS/firewall</td>
<td>416</td>
<td>22</td>
<td>46</td>
</tr>
<tr>
<td>Mac learner</td>
<td>314</td>
<td>73</td>
<td>17</td>
</tr>
<tr>
<td>Server load balance</td>
<td>951</td>
<td>145</td>
<td>34</td>
</tr>
<tr>
<td>Stateful firewall</td>
<td>None found</td>
<td>None found</td>
<td>25</td>
</tr>
<tr>
<td>Usage-based rate limiter</td>
<td>None found</td>
<td>None found</td>
<td>None found</td>
</tr>
</tbody>
</table>
• 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

![Graph showing event handling time and recompilation time for single and multi event cases. The x-axis represents event arrival rate (events/second), and the y-axis represents time (ms) on a logarithmic scale. The legend indicates different time categories: Event handling time, Single, and Recompilation time, Multi. The graph highlights the increase in time with increasing event rates.]
KINETIC: REAL DEPLOYMENTS

• 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
  – Presented in ACM CHI 2015
KINETIC TAKEAWAYS

• **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

Questions?