Troubleshooting SDNs

Peyman Kazemian
Stanford University

Why SDN Troubleshooting

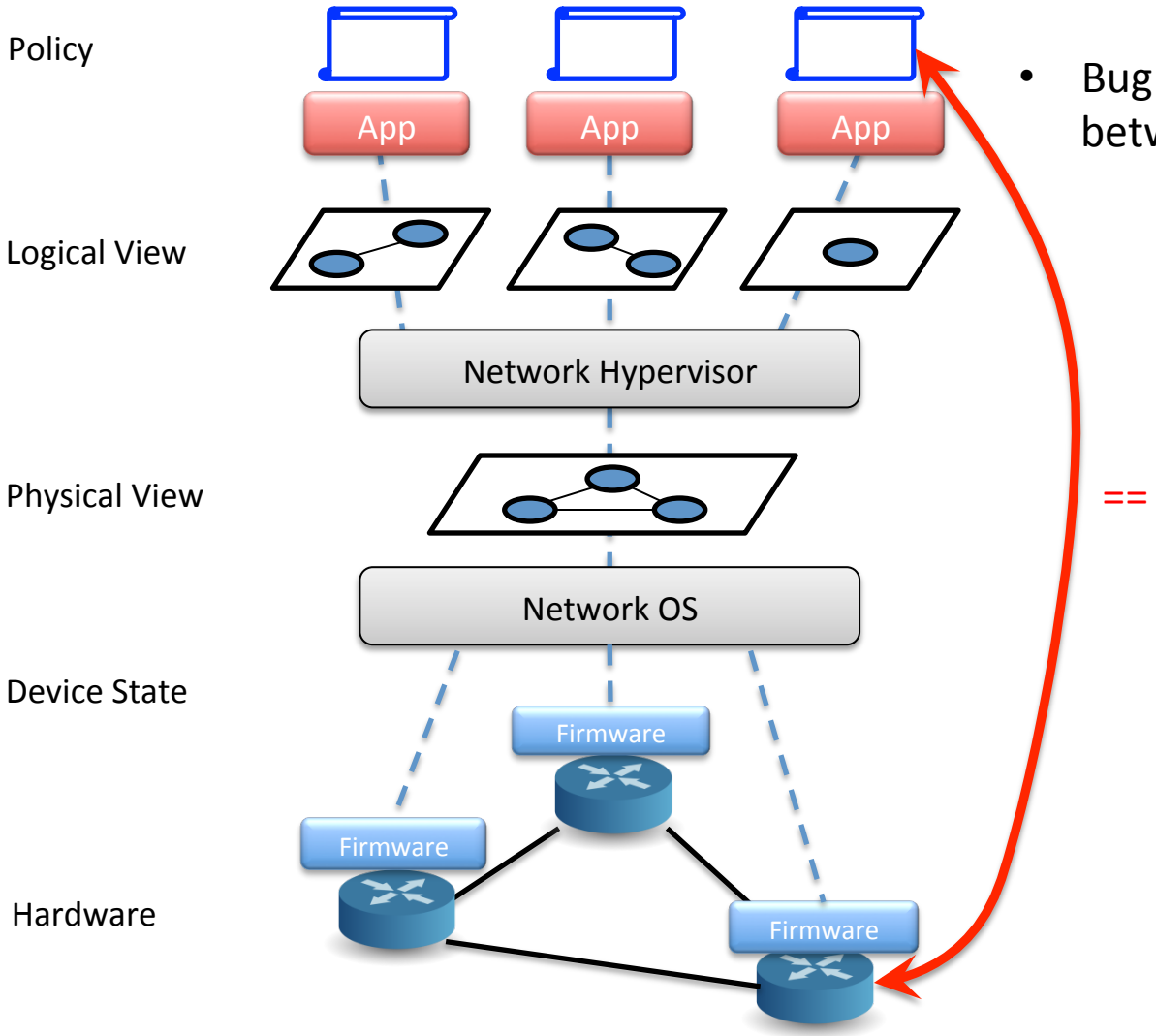
- SDN decouples software (control plane) from hardware (data plane).
  - ✓ Opens doors for innovation in networks.
  - ✓ More competition.
  - ✓ Brings down the capex.
  - ? Makes network management task easier and hence reduce opex.
    - SDN software stack is a complex distributed system working in an asynchronous environment, which introduces new bugs and troubleshooting challenges.
    - Hardware, Network OS and Apps could come from different vendors. What will happen when things break? Who to blame?

Why SDN Troubleshooting

- SDN gives us a unique opportunity for systematic troubleshooting.
  - Decouples control plane from data plane.
    - State changes pushed from a logically centralized location.
    - Easier to access/observe the state of the network.
  - SDN architecture provides clear abstraction for control plane functionality.

Richer troubleshooting techniques.

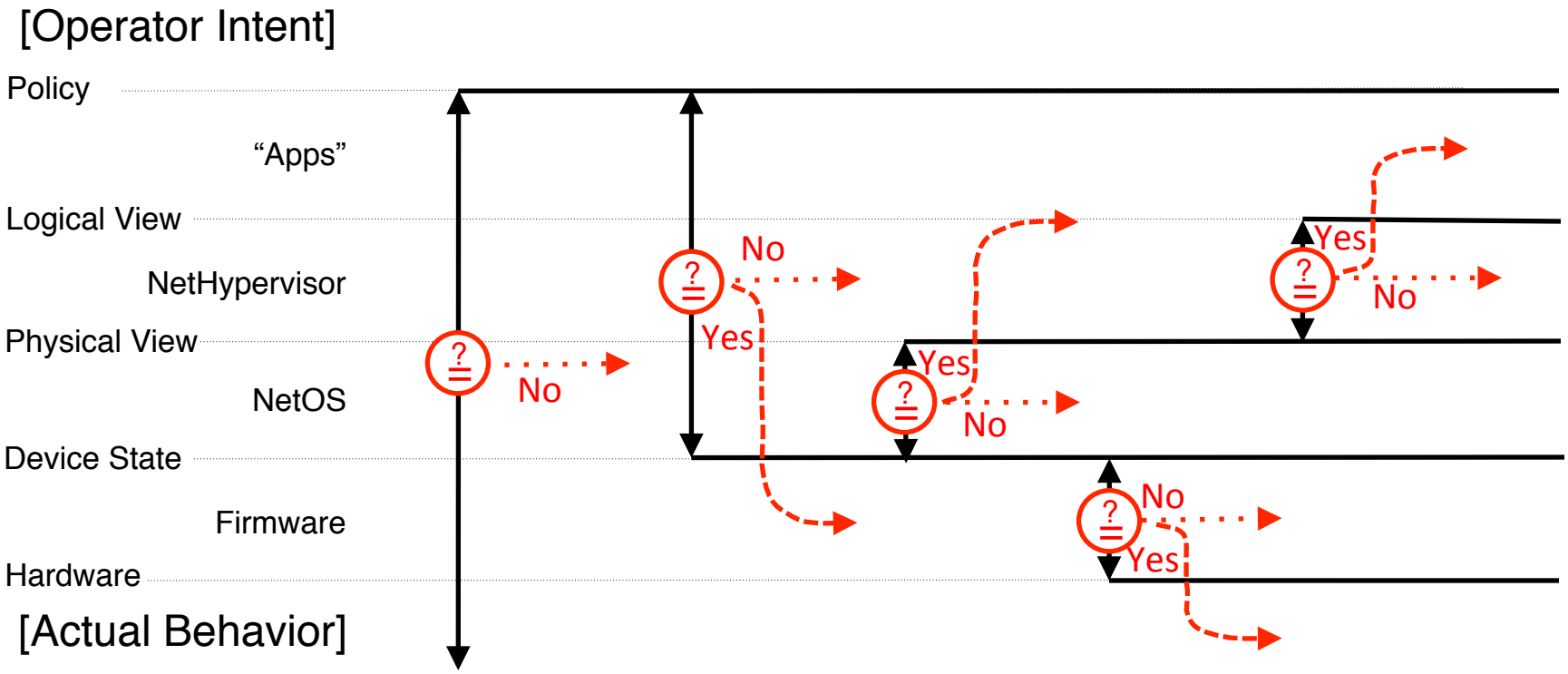
SDN Architecture



- Bug = Mistranslation between different layers

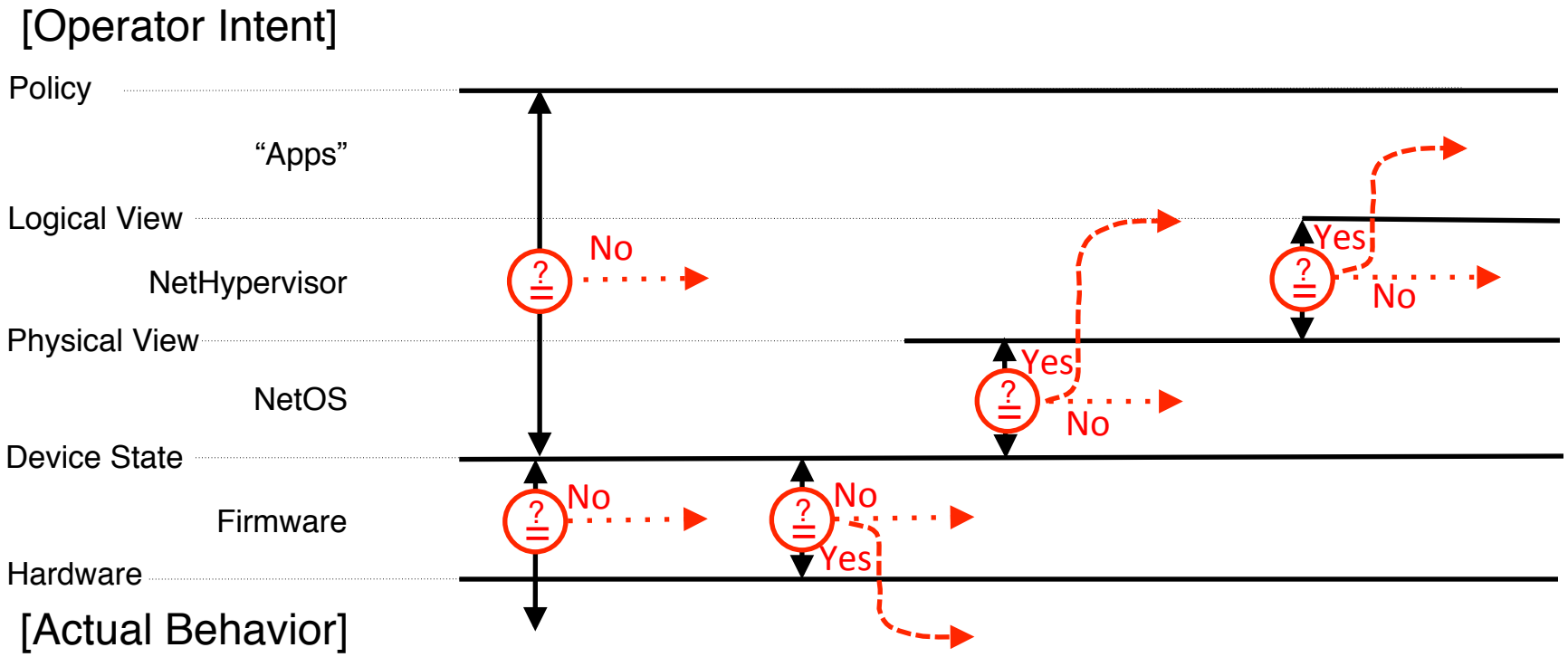
Reactive Troubleshooting of SDNs

One possible Binary Search to detect where error happens reactively.



Proactive Troubleshooting of SDNs

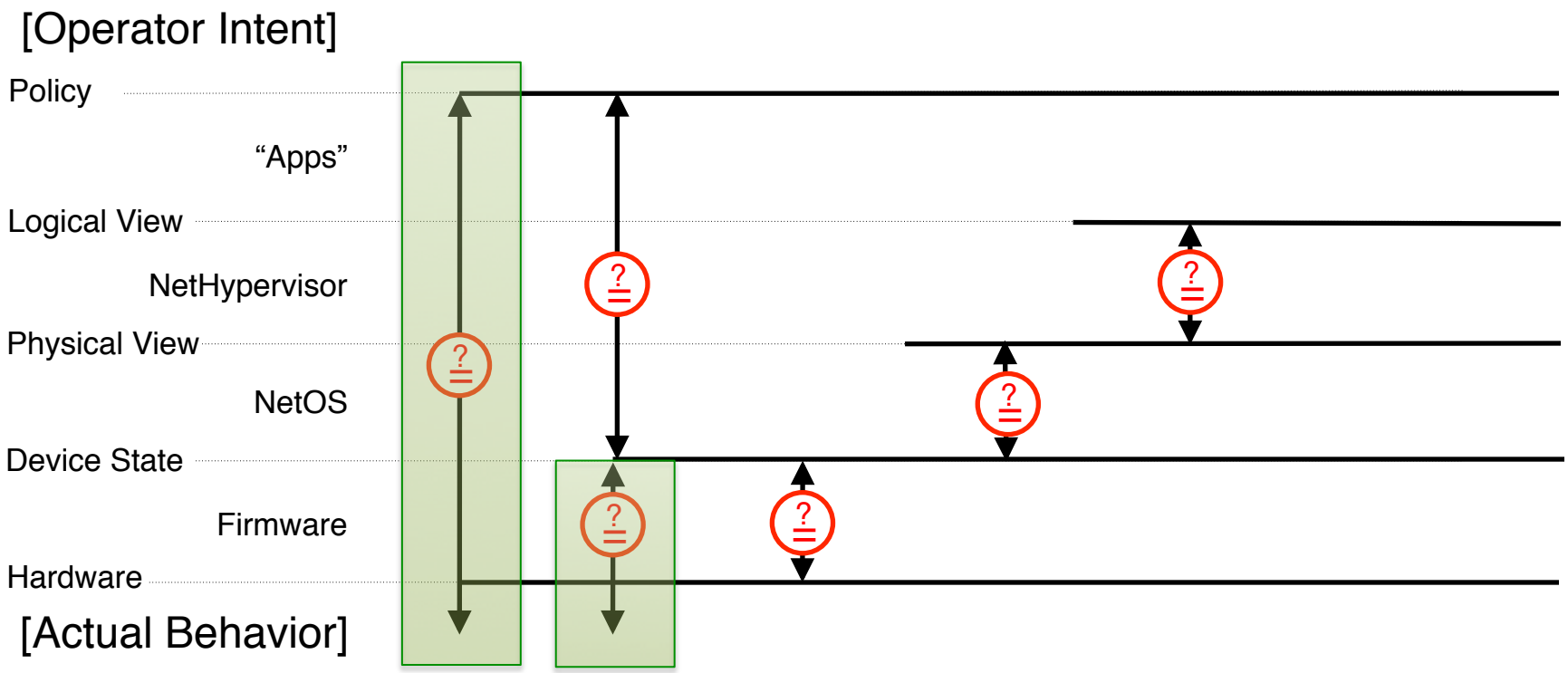
One possible Binary Search to detect where error happens proactively.



RESEARCH WORKS ON SDN TROUBLESHOOTING

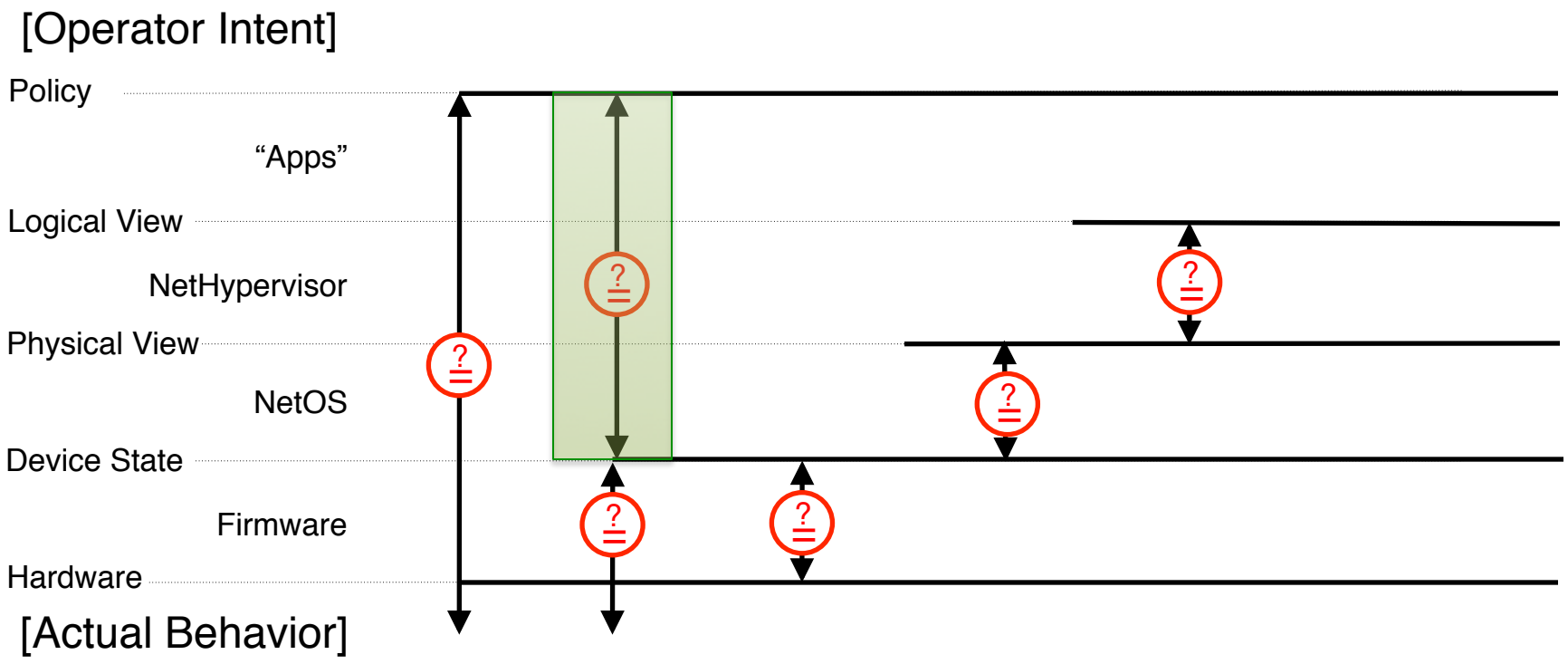
Troubleshooting SDNs

NDB (Where is the debugger for my software defined network, HotSDN'12)
ATPG: (Automatic Test Packet Generation, CoNEXT'12)



Troubleshooting SDNs

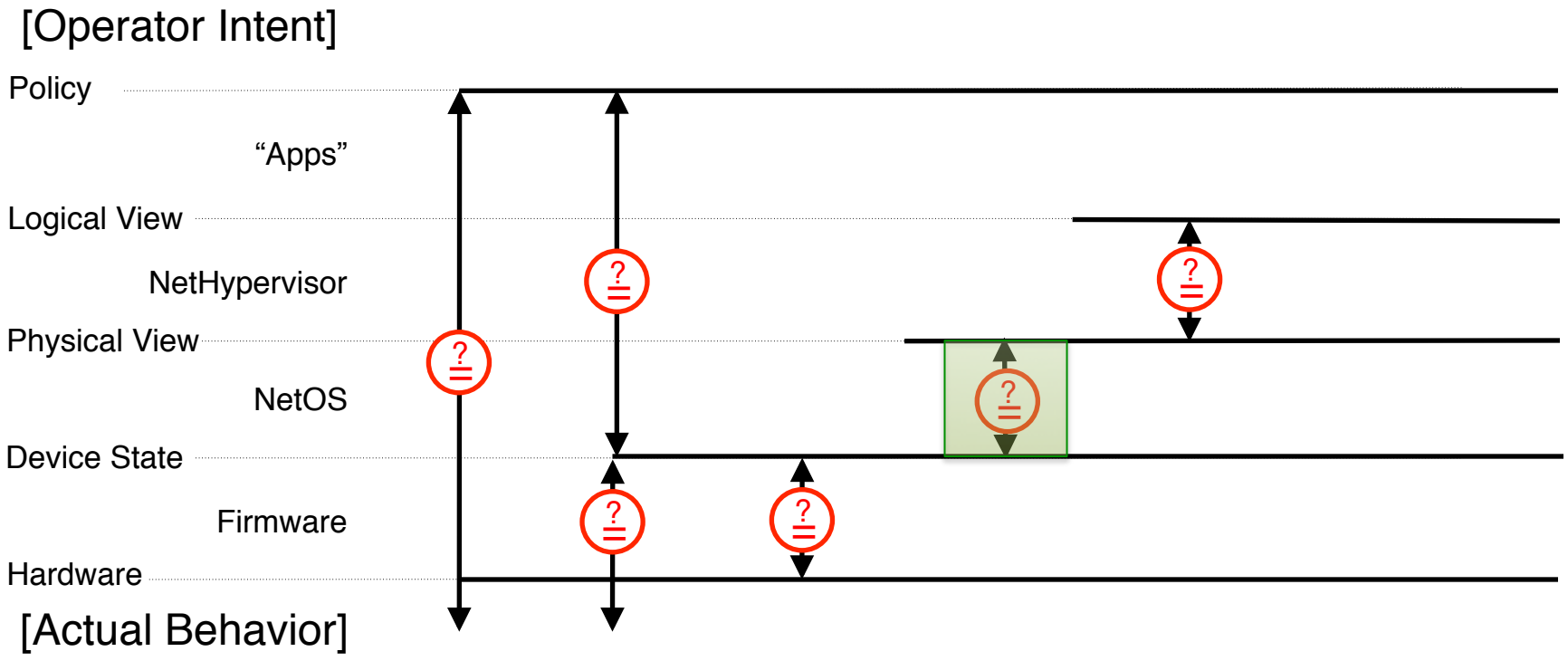
AntEater (Debugging the dataplane with AntEater, Sigcomm'11)
HSA (Header Space Analysis: static checking for networks NSDI'12)
VeriFlow (Verifying Network-wide invariants in real time, HotSDN'12)



Troubleshooting SDNs

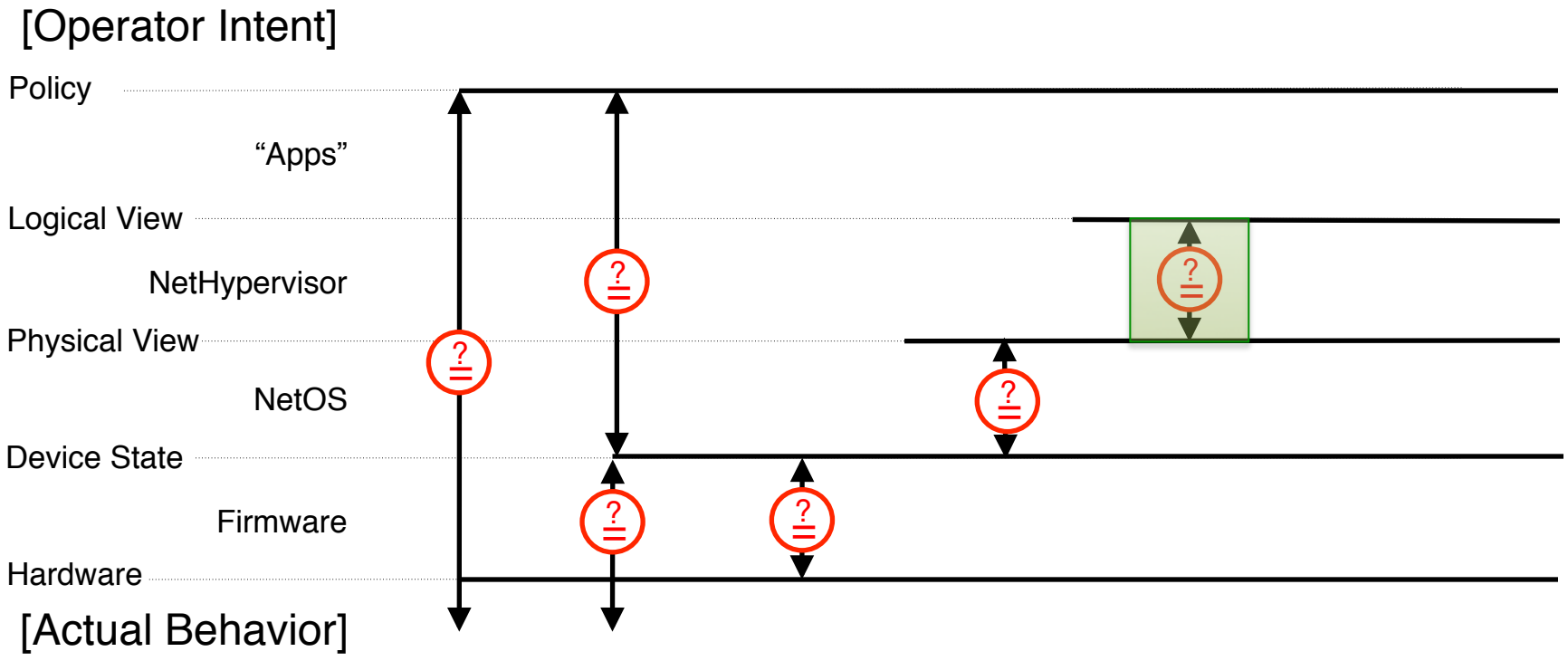
OFRewind (Enabling record and replay troubleshooting for networks, ATC'11)

NICE (a NICE way to test OpenFlow applications, NSDI'12)



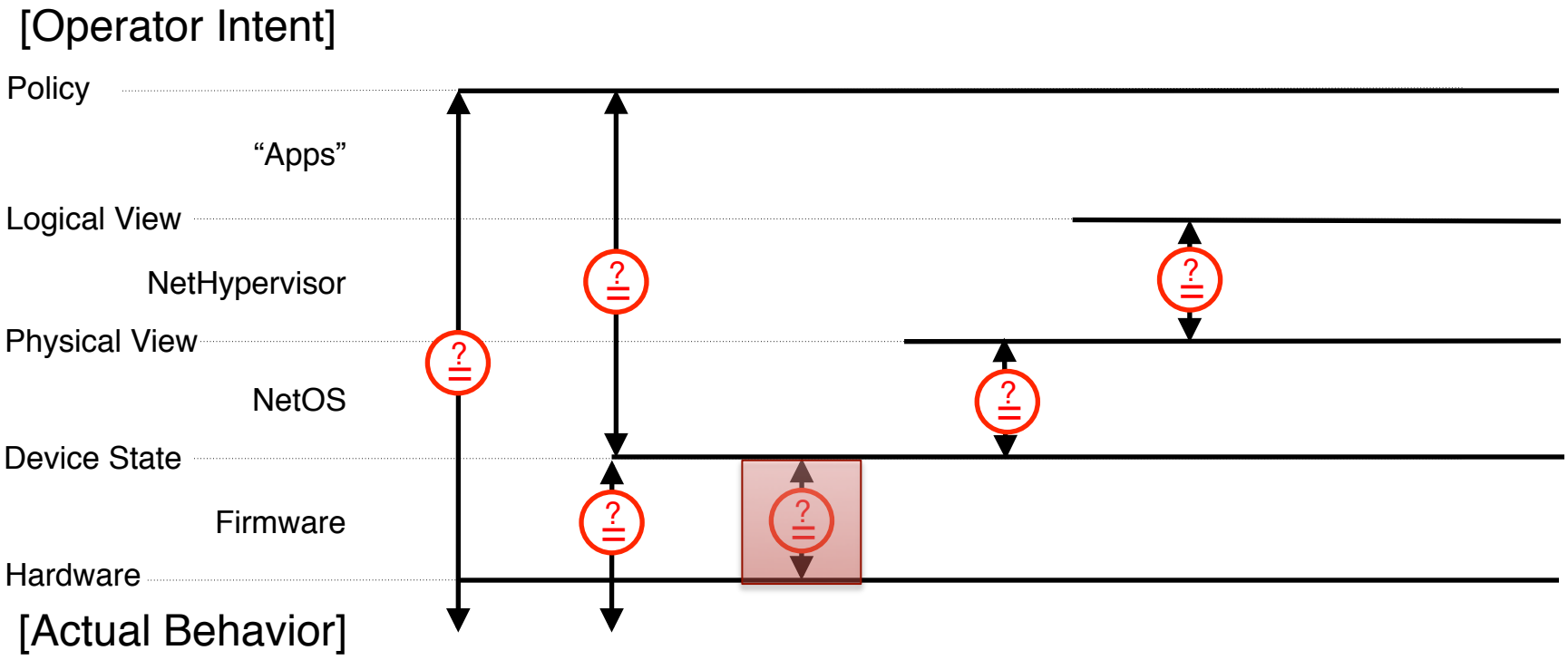
Troubleshooting SDNs

Bi-Simulation (What, Where and When: Software Fault localization for SNDs, UC Berkeley tech report)



Troubleshooting SDNs

RIB == FIB? Compare device state against the actual bits and bytes in TCAMs, etc.



WHAT ELSE IS NEEDED?

Policy Expression Language

- Rarely the policies are maintained anywhere, except in the mind of network admins!
- Systematic troubleshooting requires such clear policy description.

➢ Easy-to-use, expressive and standard network policy description language.

Better Troubleshooting Tools

- Not just detect where the problem is, but also find its root cause -- automaticaly.
  - Some of these tools can partially do that.
- Challenges:
  - What Information is needed?
    - Packet history (NDB)?
    - Control message history (OFRewind)?
    - “Logic” behind control/data plane?
    - …
  - What is the expected output?
    - The sequence of events that lead to the error?
    - The exact (relevant) state of control software and hardware?
    - Looks like a mix of networking and symbolic execution and formal verification.

Automated Troubleshooting

- Automatically run the search through different layers to pinpoint the error.
  - Example: a complete system could do
    - Real time monitoring of data plane with test packets.
    - Real time checking of network policy against control messages.
    - Problem in data plane (e.g. link down, congestion, etc)
      - Report it to a control application to reroute traffic around the troubled area.
    - Problem in control plane
      - Prevent the change from hitting data plane.

Policy Driven SDN

- Use these techniques in reverse – try to derive correct state/configurations from the policy.
- Challenges:
  - A policy can be implemented in zillion ways. How to reduce the search space?
  - Avoid conflicting implementation.
  - What is the correct level of human involvement?

Thank You!

References

- A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann. OFRewind: enabling record and replay troubleshooting for networks. In *Proceedings of USENIX ATC 2011*
- H. Zeng, P. Kazemian, G. Varghese, and N. McKeown. Automatic Test Packet Generation. In *Proceedings of CoNEXT 2012*, Nice, France, December 2012.
- P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: static checking for networks. In *Proceedings of NSDI 12,* 2012.
- M.Canini, D. Venzano, P. Perešíni, D.Kostić and J. Rexford. A NICE way to test openflow applications. In *Proceedings of NSDI 2012*
- H. Mai, A. Khurshid, R. Agarwal and S. Khurshid Agarwal, M. Caesar, P. B. Godfrey, and S. T. King. Debugging the data plane with anteater. In *Proceedings of SIGCOMM 2011*
- N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown. Where is the debugger for my software-defined network? In *Proceedings of HotSDN 2012*
- P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. Real time network policy checking using header space analysis. In *Proceedings of NSDI 2013*
- M. Kuzniar, P. Perešíni, N. Vasić, M. Canini, and D. Kostić. Automatically testing software-defined networks. In *Proceedings of HotSDN 2012*