Validating the System Behavior of Large-Scale Networked Computers

Chen-Nee Chuah

Robust & Ubiquitous Networking (RUBINET) Lab

http://www.ece.ucdavis.edu/rubinet

Electrical & Computer Engineering

University of California, Davis

Networked Computers: Some Observations

- **Different capabilities/constraints**
  - **Getting smaller & getting bigger**
- **Different requirements**
- **Explosive growth in numbers**

Wireless Sensors

Hand-held devices (PDAs, etc)

Smart home appliances

Laptops

Super computer

Intelligent transportation system

Rover Mars Vehicle

We know how individual component/layer behaves, but when they are inter-connected and start interacting with each other, we become clueless!

What do we care about when we design networks?

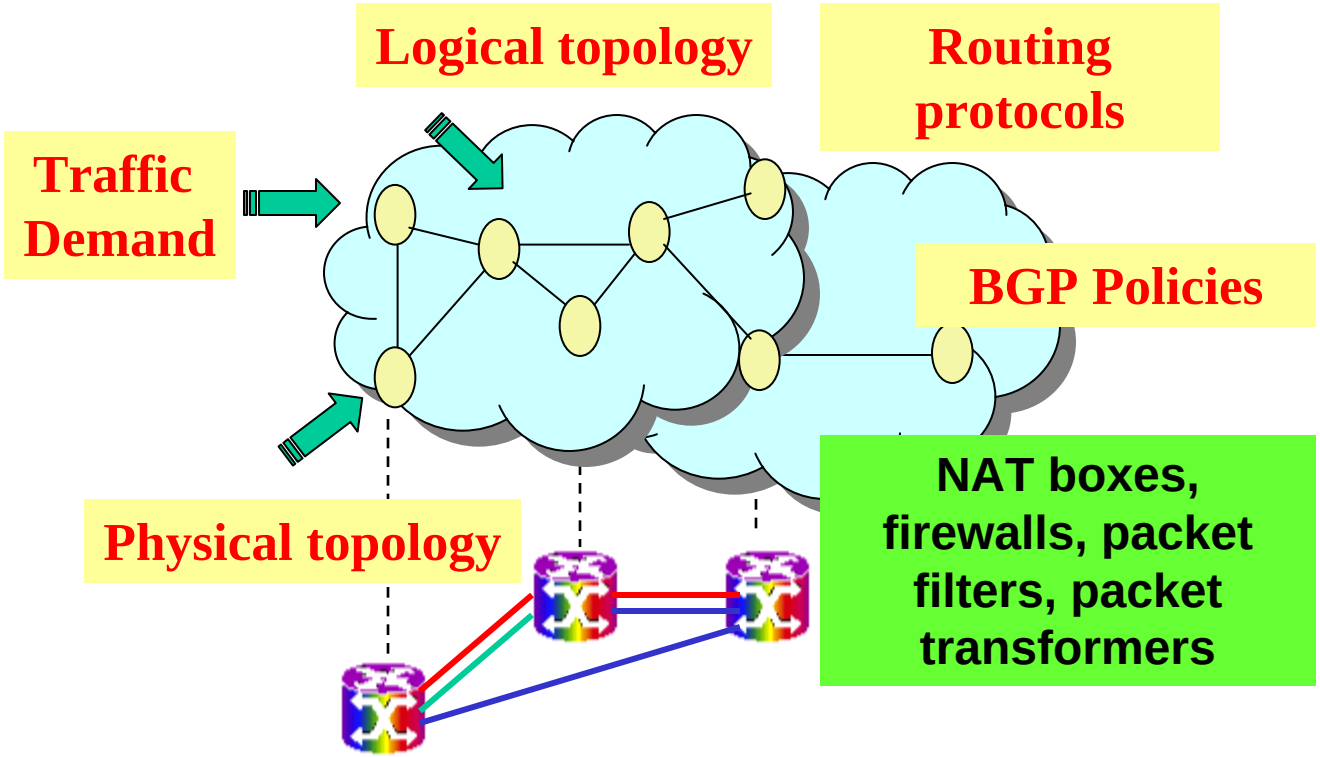
- End-to-end behavior
  - Reachability
  - Performance in terms of delay, losses, throughput
  - Security
  - Stability/fault-resilience of the end-to-end path
  - …
- System-wide behavior
  - Load distribution within a domain
  - Stability/Robustness/Survivability
  - Manageability
  - Evolvability and other X-ities
    - J. Kurose, INFOCOM'04 Keynote Speech

How do we know when we get there?

- We know how to do the following fairly well:
  - Prove correctness/completeness of stand-alone system or protocol
    - E.g., algorithm complexity, convergence behavior
  - Look at steady state, worst-case, and average scenario
    - E.g., Queuing models
  - Run simulations/experiments to show improvement of protocol/architecture Z over A, B, C, D ….
- What is lacking:
  - **End-to-end Validation** of the design solution or system behavior
    - Is the system behavior what we really intended?
    - How do we verify what type of behaviors/properties are ‘correct’ and what are ‘abnormal’?
  - **Verification of the system ‘dynamics’**, e.g., how different components or network layers interact

Challenges

- End-to-end system behavior depends on:



- Messy dependency graphs => A lot to model if we truly want to understand and able to validate system behavior

Problem Areas

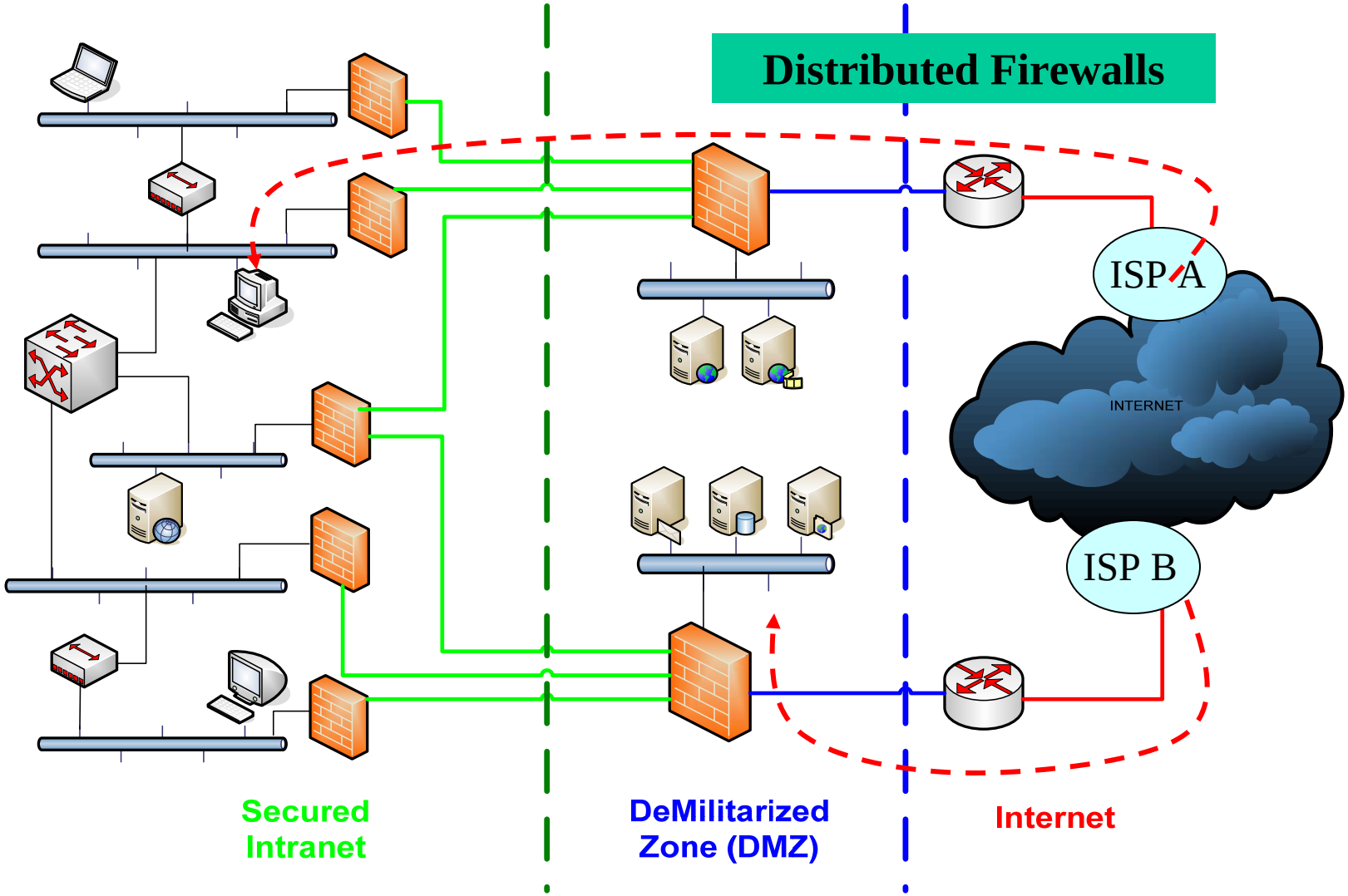
Validating

1. End-to-end network properties
   - Example: end-to-end reachability and/or security
2. Interactions between multiple control loops (across protocol layers or between multiple entities)
   - Example: overlay/IP-layer routing
3. Measurement/monitoring methodologies
   - How do we know we're measuring the traffic features that are really important instead of distorting them?

End-to-End Reachability/Security

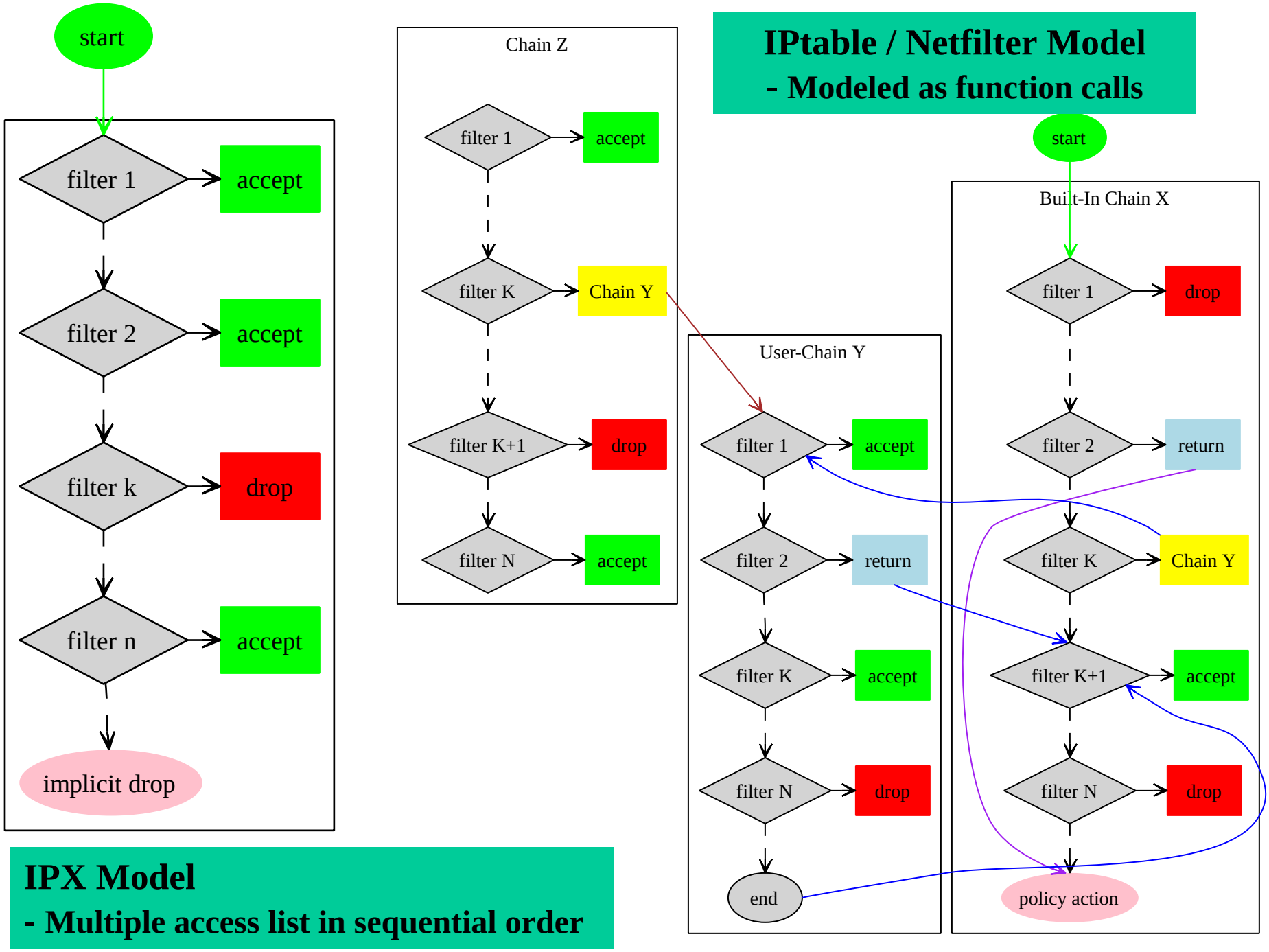
- When user *A* sends a packet from a source node *S* to a destination node *D* in the Internet
  - *How do we verify there is indeed a route that exist between S and D?*
  - *How do we verify that the packet follow a certain path that adheres to inter-domain peering relationships?*
  - *How do we verify that only this end-to-end connection satisfy some higher-level security policy?*
    - *E.g. Only user A can reach D and other users are blocked?*
- Answer depends on:
  - Router configurations & BGP policies
  - Packet filters along the way: Firewalls, NAT boxes, etc.

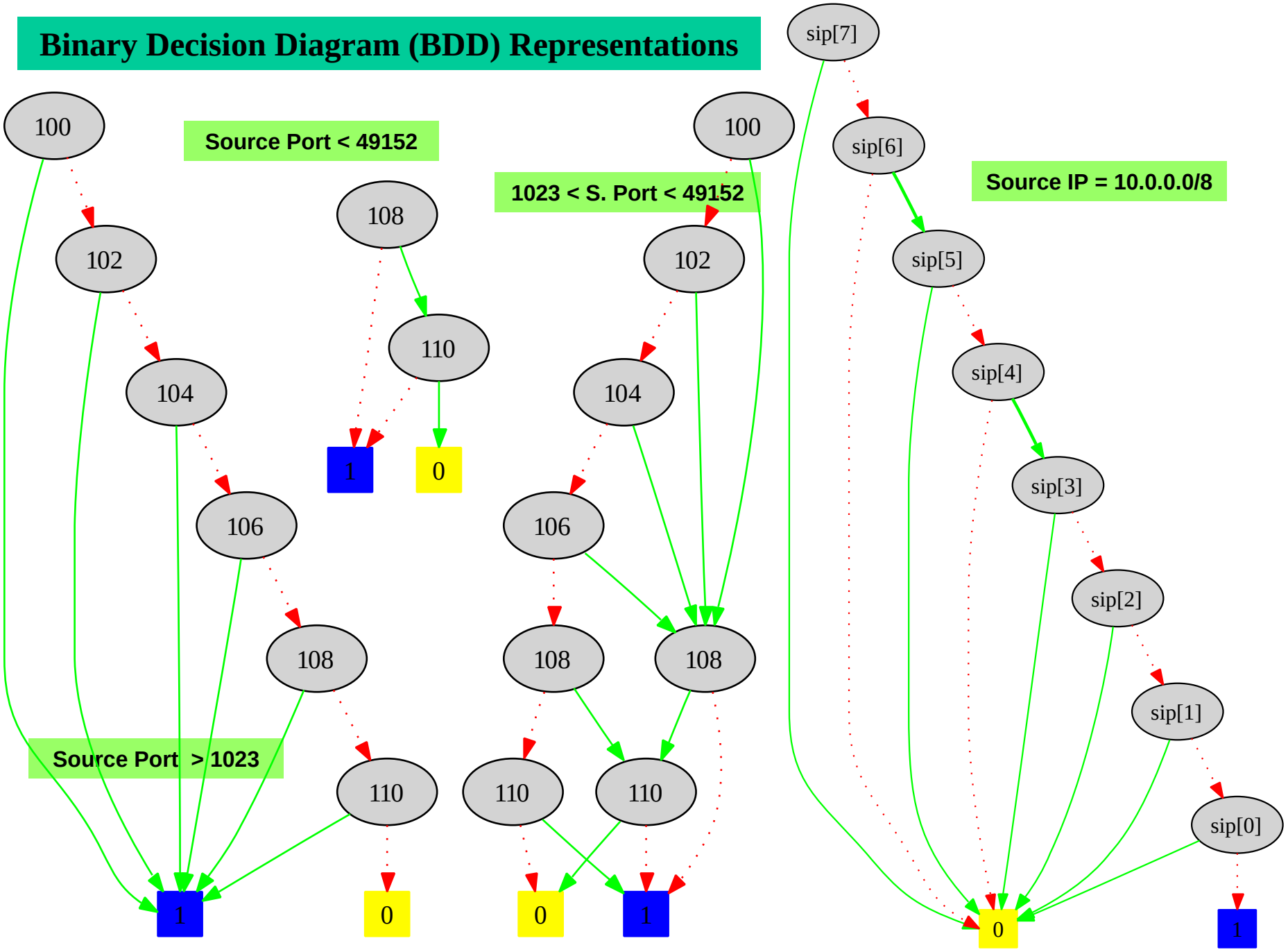
Example: Network of Firewalls



Validating End-to-End Reachability/Security

- Effectiveness of firewalls depend on (mis)configuration!
  - Policy violation
  - Inconsistency: shadowing, generalization, …
- How do we verify configuration of firewall rules?
  - Borrow model checking techniques from software programming
- Example static analysis approach
  - Control flow analysis: possible flow path
  - Data flow analysis: catching anomalies
  - Binary Decision Diagram (BDD) representations


start
filter 1
accept
filter 2
accept
filter k
drop
filter n
accept
implicit drop
Chain Z
filter 1
accept
filter K
Chain Y
filter K+1
drop
filter N
accept
IPtable / Netfilter Model
- Modeled as function calls
start
Built-In Chain X
filter 1
drop
filter 2
return
filter K
Chain Y
filter K+1
accept
filter N
drop
policy action
User-Chain Y
filter 1
accept
filter 2
return
filter K
accept
filter N
drop
end
IPX Model
- Multiple access list in sequential order


Binary Decision Diagram (BDD) Representations
Source Port < 49152
Source Port > 1023
1023 < S. Port < 49152
Source IP = 10.0.0.0/8
100
102
104
106
108
110
1
0
sip[7]
sip[6]
sip[5]
sip[4]
sip[3]
sip[2]
sip[1]
sip[0]

Network of Firewalls: Remaining Issues

- How do we validate/verify dynamic behavioral changes?
  - With multi-homing and dynamic load-balancing, the end-to-end path and sequence of firewalls traversed could change over time
  - Adaptation of firewall rules on demand depending on applications
- How do we optimize firewall configurations?
  - Inter-firewall & inter-path optimization
    - Must interface with routing plane
  - Heavy traffic ‘accepted’ first?
    - Need to interact with traffic measurement/monitoring modules

#2: Interaction btw Multiple Control Loops

Example 1: Overlay/IP-layer Interactions

- Overlays compete with IP-layer to control routing decisions
  - ISPs & overlays are unaware of decisions made by the other layer
  - Multiple overlay networks co-exist and make independent decisions
- Side Effects

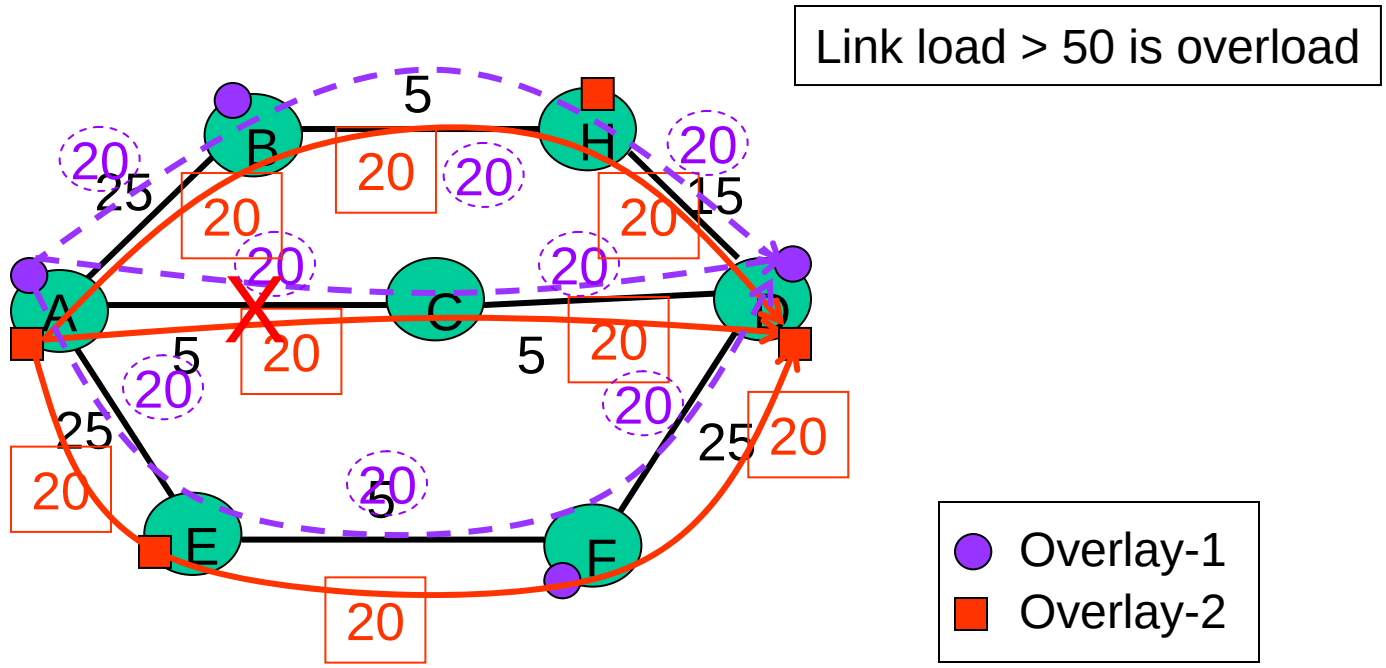
  (a) Challenges to ISP's Traffic engineering (TE)
    - Overlays *shift* and/or *duplicate* TM values, increasing the dynamic nature of the TM, making it harder to estimate
    - Harder to estimate Traffic Matrix (TM) essential for most TE tasks.

  (b) Multiple overlays can get synchronized
    - Interfere with load balancing or failure restoration, leading to oscillations

  (c) Coupling of multiple ASes
    - Overlay Networks may respond to failures in an AS by shifting traffic in upstream AS.

(b) Race Conditions & Load Oscillations

- Multiple overlays can get synchronized!



- Result of
  - Periodic nature of path probing process
  - Partial/full overlap of primary and alternate paths
- Could happen in real networks

Insights from Economic & Social Foundations

Related Studies

- Qiu et al investigate the performance of selfish routing of multiple co-existing overlays [QYZ03]
  - Optimal average latency is achieved at the cost of overloading some links
- Liu et al model interaction between IP traffic engineering and overlay routing as two-player game [LZ+05]

Other example problems

- Tuning IGP routing protocol parameters
  - Stability vs. Fast convergence
- TCP congestion control vs. IP traffic engineering

#3:Measurement/Monitoring Methodologies

- Network measurements/monitoring traditionally useful for network design and traffic engineering purposes
  - E.g., how to select optimal set of IGP link weights to route all OD pairs given a topology to distribute loads evenly across network.
- Increasingly important for anomaly detection & security forensics
  - E.g., online detection of DoS/DDoS attacks, worm/virus propagation, flash crowd, etc.

#3:Measurement/Monitoring Methodologies

- Challenges: high data speed, limited storage
  - 'Sampling' is typically done to reduce overhead
- Questions:
  - What is the optimal sampling rate?
  - Does sampling preserve the traffic features that are crucial for anomaly detection (in addition to volume estimation for TE)?
  - Can we sample less if we collect measurements at more points?

Summary

1. Validate end-to-end security/reachability properties
   - Example: firewall
   - Useful toolkit:
     - Model checking from software programming
     - Combinatorial optimization
2. Model system dynamics and interactions between entities
   - Example: overlay/IP-layer routing
   - Borrow economic models: game theory
3. Verify measurement/monitoring methodologies
   - How do we know we're measuring the traffic features that are really important instead of distorting them?