From Electronic Design Automation to NDA: Treating Networks like Chips or Programs

George Varghese
With Collaborators at Berkeley, Cisco, MSR, Stanford
Networks today

- **Multiple Protocols**: 6000 RFCs (MPLS, GRE . . .)
- **Multiple Vendors**: Broadcom, Arista, Cisco, . . .
- **Manual Configurations**: Additional arcane programs kept working by “masters of complexity” (Shenker)
- **Crude tools**: SNMP, NetFlow, TraceRoute, . . .

![Diagram showing SQL 1001, 10* → P1, 1* → P2, Drop SQL, Load balancing, Access Control Lists (ACLs)]
Simple questions **hard** to answer today

- Which packets from A can reach B?
- Is Group X provably isolated from Group Y?
- Is the network causing **poor performance** or the server?
- Why is my backbone **utilization** poor?
- Is my load balancer **distributing** evenly?
- Where are there mysterious packet losses?

**BOTTOM UP ANALYSIS OF EXISTING SYSTEMS**
Motivation to do better

- **Internal:**
  - Errors often caused by configuration changes

- **External:** (2012 NANOG Network Operator Survey):
  - 35% > 25 tickets per month, > 1 hour to resolve
  - Welsh: vast majority of Google “production failures” due to “bugs in configuration settings”

As we migrate to services ($100B public cloud market), network failure will be a debilitating cost.
Networks Tomorrow

- Online services → latency, cost sensitive
- Merchant Silicon → Build your own router
- Rise of Data centers → Custom networks
- Software defined Networks (SDNs) → custom design “routing program”
- P4 (next generation SDN) → redefine hardware forwarding at runtime

TOP DOWN DESIGN OF FUTURE NETWORKS TO
OPTIMIZE GOAL
Digital Hardware Design as Inspiration?

Electronic Design Automation (McKeown SIGCOMM 2012)

Network Design Automation (NDA)?
Outline

• Part 1: Tools for operators *today*
  o Static Analysis, Test Packet Generation
  o *Analysis* via Symbolic Execution

• Part 2: Tools, processes for designers & operators *tomorrow.*
  o Network Design Automation
  o *Synthesis* via Optimization
Many forwarding flavors / 1 essence

IP Router

10* → P1
1* → P2

MAC Bridge

01A1A2 → P1

MPLS Switch

5 → P1, Pop 5

PREFIX MATCH

EXACT MATCH

INDEXED LOOKUP

ESSENTIAL INSIGHT FOR OPENFLOW. USE SAME INSIGHT FOR UNDERSTANDING EXISTING PROTOCOLS
Idea: Treat Network as a Program

- Model header as point in high dimensional space and all networking boxes as transformers of header space

NETWORK BOX ABSTRACTED AS SET OF GUARDED COMMANDS . .
NETWORK BECOMES A PROGRAM ➔ CAN USE PL METHODS
Header Space Framework

- Model all networking boxes as transformers of header space

Transfer Function:

\[ T : (h, p) \rightarrow \{(h_1, p_1), \ldots, (h_n, p_n)\} \]
Computing Reachability (Kazemian et al, NSDI 12)

- As in hardware, *automatically* generate test packets to detect faults
- Different optimization from hardware testing:
  - Maximize link/queue coverage
  - Performance (e.g., latency) not stuck-at faults
  - Respect constraints on terminal ports
- Up to 160X reduction over all-pairs - aspects in Microsoft Autopilot
- Bounded network graph allows simple set cover compared to program testing (KLEE)
Semantics (Plotkin et al)

New semantics that has:

- **Symmetry Theorem**: Can reduce fat tree to "thin tree" using a "simulation" and verify reachability cheaply in latter
- **Modularity Theorem**: Reuse of parts of switching network
Tool 4: Batfish (Fogel et al, NSDI ‘05)

• So far all tools are for network data plane
• Need control plane tools for proactive analysis
  • Check configuration sanity before applying to the network
  • Check safety in the presence of certain routing changes
  • Check back-ups are properly implemented
Other Work

- Geometric Packet Classification. (SIGCOMM 1998)
- Static Reachability of IP Networks (INFOCOM 2005)
- Anteater. (SIGCOMM 2011)
- Veriflow. (HotSDN 2012)
- SAT Based Data Plane Verification (HotSDN 2012)
- Flowlog (HotSDN 2012)
- NetKat/Netcore
PART 2: NETWORK SYNTHESIS VIA OPTIMIZATION
Network Design Automation?

HOW MIGHT WE GO BEYOND EARLY WORK? WHAT NEW AREAS CAN WE TOUCH?
Static Checkers: Booleans \(\rightarrow\) Quantities

- Given end-to-end flow rates, calculate link loads in face of failures (Juniwal et al, in progress)
- Given flow rate histograms, pack as many flows as possible & keep overflow probability within threshold
Probabilistic Knapsack – (w. Bjorner, Gopalan, Karp, Kannan) Packing distributions

- **Correctness**: Failure probability < T, e.g., T = 0.05
- **Performance**: Find subset that *minimizes* expected waste

Likely hard: even checking if one subset fails is exponential
Other Synthesis Problems

• **Synthesizing Rules:** Synthesize ACLs based on policy (Kang et al, Princeton)

• **Synthesizing Virtual Networks:** Rao et al (Purdue) & Xie et al (Princeton)

• **Synthesizing Tables within a router:** Table Synthesis P4 Routers (Jose et al, Stanford)

• **Synthesizing Transports:** Deadline driven alternatives to TCP (MSR Cambridge)
Interactive Debugging (AEV 15)

- Existing network debuggers (MSR Sherlock, Stanford NDB, Berkeley Xtrace are Batch Debuggers)
- What might equivalent be of setting a Watch point and then “stepping into” network?
- **Example:** Stepping Into by New Trace route message. Old TraceRoute not real-time. New hardware
## Exploiting Domain Structure

<table>
<thead>
<tr>
<th>Technique</th>
<th>Structure exploited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header Space Analysis (Symbolic Execution)</td>
<td>Limited negation, no loops, small equivalence classes</td>
</tr>
<tr>
<td>Net Plumber (Incremental Verification)</td>
<td>Network Graph, rule dependencies structure</td>
</tr>
<tr>
<td>ATPG (Test Generation)</td>
<td>Network graph limits size of state space compared to KLEE</td>
</tr>
<tr>
<td>Exploiting Symmetry</td>
<td>Known symmetries because of design (vs on logical structures)</td>
</tr>
</tbody>
</table>
Conclusion

• **Inflection Point:** Rise of services, data centers, Software Defined Networks
• **Ideas:** Symbolic execution (analysis) & optimization (synthesis)
• **Intellectual Opportunity:** Rethink existing techniques exploiting domain structure
• **Systems Opportunity** Working chips with billions of gates Why not large operational networks next?
Collaborators

MSR: Nikolaj Bjorner, Patrice Godefroid, Karthick Jayaram Nuno Lopes, Ratul Mahajan. Ming Zhang
Stanford: Peyman Kazemian, Nick McKeown, James Zheng
Berkeley: Garvit Juniwal, Sanjit Seshia
Cisco: Mohammed Alizadeh, Tom Edsall