A Theorem Proving Approach Towards Declarative Networking

Anduo Wang\textsuperscript{1} Boon Thau Loo\textsuperscript{1}
Changbin Liu\textsuperscript{1} Oleg Sokolsky\textsuperscript{1} Prithwish Basu\textsuperscript{2}

\textsuperscript{1}University of Pennsylvania
http://netdb.cis.upenn.edu/fvn/

\textsuperscript{2}BBN technologies

TPHOLs emerging trends, 2009
Motivation

- Challenges to today’s Internet: increasing complexity and fragility in Internet routing
- Growing interest in the formal verification of network protocol design and implementation
Motivation

- Challenges to today’s Internet: increasing complexity and fragility in Internet routing
- Growing interest in the formal verification of network protocol design and implementation
- Correct-by-construction, Metarouting algebraic models
  - Idealized model unlikely to be adapted to actual implementation
- Runtime verification, model checking: CMC, MaceMC, PiP, D3S
  - Inconclusive and restricted to small network
Motivation

- Challenges to today’s Internet: increasing complexity and fragility in Internet routing
- Growing interest in the formal verification of network protocol design and implementation
- Correct-by-construction, Metarouting algebraic models
  - Idealized model unlikely to be adapted to actual implementation
- Runtime verification, model checking: CMC, MaceMC, PiP, D3S
  - Inconclusive and restricted to small network
- We propose Formally Verifiable Networking framework
  - Bridge the gap between verification and design/implementation
Our Approach: *Formally Verifiable Networking*

► **Goal**: unifying the design, specification, implementation, and verification of networking protocols within a logic-based framework

► **Declarative networking** written in Network Datalog (NDlog, distributed variant of Datalog), intermediary layer between logical specification and real implementation

► Property preserving translations, from declarative networking implementation to formal system specifications for verification

► Theorem prover, statically checks properties of formal system specification against network invariants

► Code generation from verified formal specification
Background on Declarative Network

- Declarative specifications of networks using *Network Datalog* (NDlog), a distributed variant of Datalog
- NDLog is compiled to distributed dataflows
- Distributed query processor executes the dataflows to implement the network protocols

\[
R1: \text{reachable}(@S,D) \leftarrow \text{link}(@S,D)
\]
\[
R2: \text{reachable}(@S,D) \leftarrow \text{link}(@S,Z), \text{reachable}(@Z,D)
\]

More details on declarative networking

- Additional publications: Sigcomm05, Sigmod06, SOSP05
Background on Declarative Network

- Declarative specifications of networks using Network Datalog (NDlog), a distributed variant of Datalog
- NDLog is compiled to distributed dataflows
- Distributed query processor executes the dataflows to implement the network protocols

\[ \text{R1: reachable}(S,D) \leftarrow \text{link}(S,D) \]
\[ \text{R2: reachable}(S,D) \leftarrow \text{link}(S,Z), \text{reachable}(Z,D) \]

- For all nodes S,D: S can reach D via a link from S to D

More details on declarative networking

- Additional publications: Sigcomm05, Sigmod06, SOSP05
Background on Declarative Network

- Declarative specifications of networks using *Network Datalog* (NDlog), a distributed variant of Datalog
- NDLog is compiled to distributed dataflows
- Distributed query processor executes the dataflows to implement the network protocols

R1: \( \text{reachable}(\text{S}, \text{D}) \leftarrow \text{link}(\text{S}, \text{D}) \)

R2: \( \text{reachable}(\text{S}, \text{D}) \leftarrow \text{link}(\text{S}, \text{Z}), \text{reachable}(\text{Z}, \text{D}) \)

- For all nodes \( \text{S, D, Z} \): if there is a link from \( \text{S} \) to \( \text{Z} \), and that \( \text{Z} \) can reach \( \text{D} \), then \( \text{S} \) can reach \( \text{D} \)

More details on declarative networking

  Additional publications: *Sigcomm05*, *Sigmod06*, *SOSP05*
FVN Framework

- **Specification**: two way property preserving translation
  - Formal system specification generated from NDlog program (arc 4)
  - Declarative network synthesized from verified logical specification (arc 3)
- **Verification**: proving network invariants against system specifications by interacting in theorem prover
- **Implementation**: existing distributed query processor executes the NDlog programs
Conclusion

- More details in poster
  - NDlog Program Verification (arc 1,4,5)
  - Generating Equivalent NDlog implementation (arc 3,7)
  - Component Based Verification of BGP System (arc 2,3,5)

- Future Work
  - Network Models and Implementation
    - Relaxed algebraic models, component-based models
  - Modeling Soft-state in Declarative Networking
    - Linear logic, semantic foundation for verification of NDlog programs with soft-state
  - Combining Verification Techniques
    - Theorem proving, declarative networking specific proof strategies/tactics
    - Model checking declarative networking, transitions update routing tables specified in linear logic