Formally Verifiable Networking

Anduo Wang¹, Limin Jia¹, Changbin Liu¹, Boon Thau Loo¹, Oleg Sokolsky², Prithwish Basu²
¹Department of Computer and Information Science, University of Pennsylvania ²BBN technologies

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 formal model unlikely to be adapted to actual implementation
- Runtime verification, programming framework, model checking: CMC, MaceMC
- Inconclusive and restricted to small network
- We propose Formally Verifiable Networking
- Bridge the gap between verification and design/implementation

Formally Verifiable Networking
- Unifying the design, specification, implementation, and verification of networking protocols.
- Formal/logical statements specify the behavior and the properties of the protocol
- Theorem proving establishes correctness of formal system specification wrt network properties
- Declarative networking, intermediary layer between logical specification and implementation
- Property preserving translations from declarative networking implementation to formal system specifications for verification
- Code generation from theorem prover
- Meta-model, correctness-by-construction design

Verified Code Generation
- Component Based Verification of BGP System
- Specification of BGP components in PVS
- Correctness proofs in PVS
- Metatheoretic verification

Abstract Routing Algebra in PVS
- routeAlgebra: THEORY
- BEGIN
  sig: TYPE+ label: TYPE
  injective(label = bool)
  sig: TYPE = (l: label) | inject(l)
  prohibPath: sig
  labelApply: (l, sig = sig)
  prohibRel: (sig, sig = bool)
eqRel(S, Z): sig = bool = prohibRel(S, Z) \& prohibRel(S, Z)
monotonicity: label = sig = bool = prohibRel(S, Z + prohibRel(Z, Y))
axiom: (\neg \exists (l: label) = prohibRel(s, t)) \& prohibRel(s, t)
absorption: prohibRel

Base Algebra for Shortest Path Routing
- Abstract Route Algebra
- Add: TYPE

PVS mapping generates instances of routeAlgebra axioms as Type Correctness Conditions (TCCs)

Shortest Path Routing in PVS
- routeAlgebra: Interpreted Theory: Base Algebra addA: THEORY
  BEGIN
  m: postul m: postul redundant: postul
  NaM: Axiom \ m < \ m
  LABEL: TYPE = up(t)
  SIG: TYPE = up(t + 1)
  PROOF: LABEL, SIG = IP (LABEL = SIG)
  THEN (t = 1) \ IMPLIES (LABEL = SIG)

Base Algebra for Provider-Customer, Peer-Peer Guideline
- For economical reasons, ISP reduces use of provider routes, and maximizes availability of customer routes
  \label{eq:routePath}

Provider-Customer, Peer-Peer Guideline in PVS
- For simplicity, rename labels and signatures: c = 1, r = 2, p = 3, and C = 1, R = 2, P = 3
- IP: THEORY

BGPSystem: THEORY
SIG: TYPE = up(t)
LABEL: TYPE = up(t)
IMPORTING importRouteAlgebra

Per-Route Specification
- Route paths are measured in terms of customer provider relationship and distance
- Customer-provider Peer-Peer guideline must be enforced
- Once customer-provider policy is satisfied, ISP wants least-cost (shortest) paths
- Decompose this BGP system into two sub-components

BGPsystem: THEORY

Future Work
- Network Models and Implementation
  - Related algebraic models: wide range of protocols
  - Alternative component-based models: Click, Xor
  - Modeling Soft-state in Declarative Networking
  - Soft-state specified as NDLog tuples (predicates) in tables that will timeout, message reordering and/or losses

Declarative specifications of networks using Network Datalog (NDlog), a distributed variant of Datalog
- Correct-by-construction, Metarouting algebraic models
- Growing interest in the formal verification of network protocol implementation
- Specification: two way property preserving translation
- Implementation: distributed query processing

Growing Equivalent NDlog Implementation
- Specify atomic component T as a rule taking T as rule body, deriving out(O3) as rule head
- Property preserving translations from declarative networking implementation to formal system specifications for verification
- Code generation from theorem prover
- Meta-model, correctness-by-construction design

FVN Overview
- Design: correctness-by-construction via meta-model
- Specification: two way property preserving translation
- Formal system specification generated from NDlog program (arc 4)
- Executable Declarative Network synthesized from verified logical specification (arc 3)
- Verification: proving network invariants of system specifications by interacting with theorem prover (arc 5)
- Implementation: distributed query processing (arc 7)

Background on Declarative Network
- Declarative specifications of networks using Network Datalog (NDlog), a distributed variant of Datalog
- NDlog is compiled to distributed dataflow
- Distributed query executor the dataflows to implement the network protocols

NDlog Program Verification
- Verification
- Network specification
- Design
- Specification
- Implementation

NDlog program for path-vector protocol
- Node N sends bestHopConf(C, P)
- Link L = (N, P)
- \text{phi}(N, P)
- \text{phi}(L, \text{phi}(N, P))
- \text{phi}(L, \text{phi}(L, \text{phi}(N, P)))

Routing tables and networks
- Soft state: network state expires after Time-To-Live (TTL) unless refreshed
- Ensures eventual consistency in protocol in the presence of message reordering and losses
- Additional rewrite step required for rules that uses soft-state predicates

Example Properties Verified using Soft-State
- Eventual convergence in stable network
- \text{bestHopCost}(S, D, C)
- \text{bestHopConf}(S, D, C)
- \text{bestHopConf}(S, D, C)

Overview of PVS System
- Abstract Routing Algebra
- PVS declaration
- Base Algebra
- Composition Operators

- A: uninterpreted source theory routeAlgebra
- Interpretation instantiated from A
- C: PVS theory taking routing algebra theories as parameters

Property preserving translations from declarative networking implementation to formal system specifications for verification
- Code generation from theorem prover
- Meta-model, correctness-by-construction design