Formally Verifiable Networking

Anduo Wang[1], Limin Jia[1], Changbin Liu[1], Boon Thau Loo[1], Oleg Sokolsky[1], Prithwish Basu[2]

[1]Department of Computer and Information Science, University of Pennsylvania [2]BBN technologies

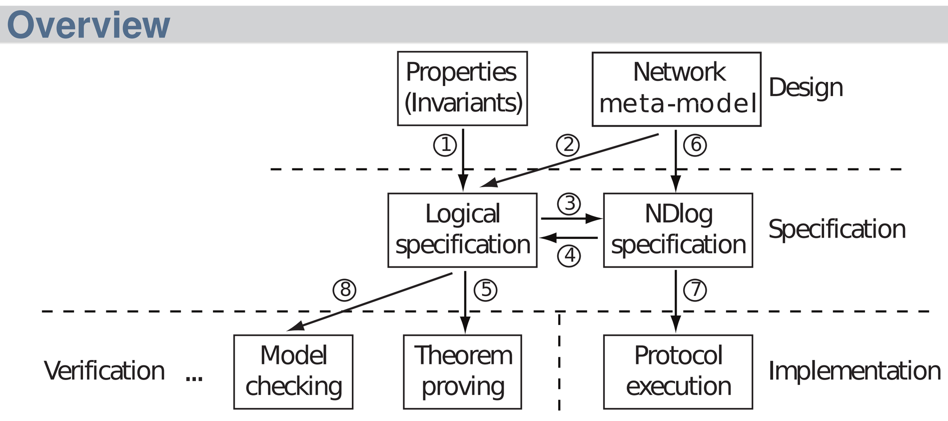
Motivation

- 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 model unlikely to be adapted to actual implementation
- Runtime verification, programming framework, model checking: CMC, MaceMC
  - Inconclusive and restricted to small network
- We propose *F*ormally *V*erifiable *N*etworking
  - *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 w.r.t network properties
- Declarative networking, intermediary layer between logical specification and real implementation
  - Property preserving translations from declarative networking implementation to formal system specifications for verification
  - Code generation from verified formal specification
- 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)

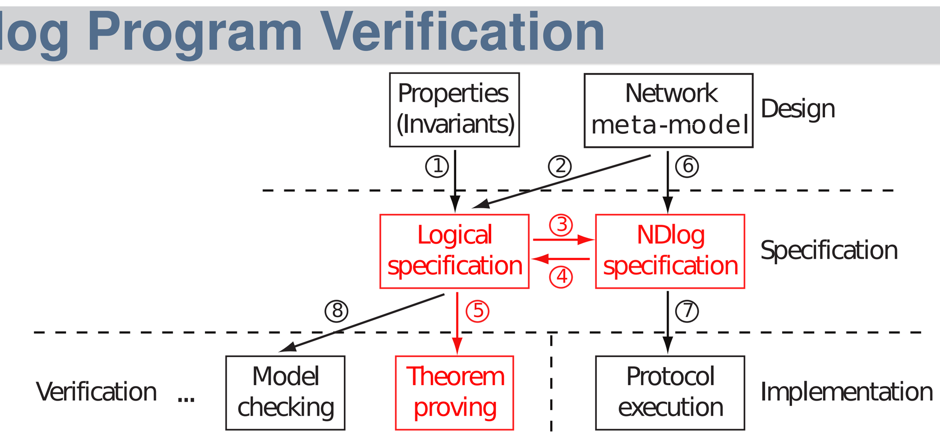
Background on Declarative Network

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

```
R1:reachable(@S,D)<-link(@S,D)
R2:reachable(@S,D)<-link(@S,Z),reachable(@Z,D)
```

- For all nodes `S,D`: `S` can reach `D` via a link from `S` to `D`
- For all nodes `S,D,Z`: if there is a link from `S` to `Z`, and that `Z` can reach `D`, then `S` can reach `D`

NDlog Program Verification



- NDlog program for path-vector protocol
  - `p1 path(@S,D,P,C):- link(@S,D,C),P=(S,D).`
  - `p2 path(@S,D,P,C):- link(@S,Z,C1), path(@Z,D,P2,C2),C=C1+C2, P=concatPath(Z,P2).`
- Encoding path-vector in theorem prover
  - p1: $\forall(S,D,P,C).link(S,D,C) \wedge P = f_{init}(S,D) \implies path(S,D,P,C)$
  - p2: $\forall(S,D,P,C).\exists(C_1,C_2,Z,P_2).link(S,Z,C_1) \wedge bestPath(Z,D,P_2,C_2) \wedge C = C_1 + C_2 \wedge P = f_{concatPath}(Z,P_2) \implies path(S,D,P,C)$
- Proving Route optimality property in PVS
  - `FORALL (S,D:Node) (C:Metric) (P:Path): bestPath(S,D,P,C) => NOT (EXISTS (C2:Metric) (P2:Path): path(S,D,P2,C2) AND C2<C)`
  - `("" (skosimp*) (expand bestPath) (prop) (expand bestPathCost) (prop) (skosimp*) (inst -2 C2!1) (grind))`

Handling soft-state in 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/or losses
- Additional rewrite step required for rules that uses soft-state predicates
  - *Declarative Network Verification.*, Anduo Wang, Prithwish Basu, Boon Thau Loo, Oleg Sokolsky., University of Pennsylvania, Technical Report No. MS-CIS-08-34, 2008

Example Properties Verified using Soft-State

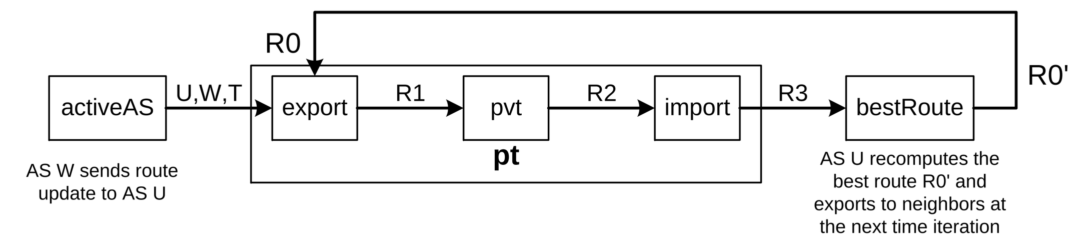
- Eventual convergence in stable network

```
bestHopCost_converge:  THEOREM EXISTS (j:posnat):
 FORALL (S,D:Node)(C:Metric)(i:posnat):  (i>j)
 => bestHopCost(S,D,C,5*i,10)
  = bestHopCost(S,D,C,5*j,10)
```

- Divergence (*count-to-infinity problem*) in dynamic network
- A well known solution: *split-horizon* can avoid count-to-infinity in two-node cycle, but cannot prevent the problem in three-node cycle

Verified Code Generation

- Component Based Verification of BGP System



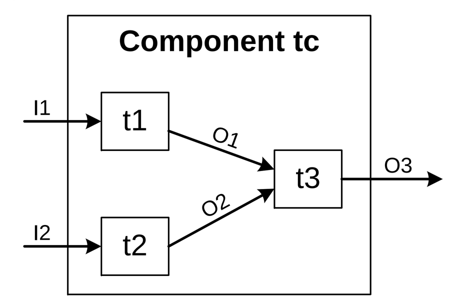
- Specification of BGP components in PVS
  - `bgp(U,W,R0,R3,T): INDUCTIVE bool = EXISTS (R1,R2): activeAS(U,W,T) AND pt(U,W,R0,R3,T) AND bestRoute(W,T,R0)`
  - `pt(U,W,R0,R3,T): INDUCTIVE bool = export(U,W,R0,R1,T) AND pvt(U,W,R1,R2,T) AND import(U,W,R2,R3,T)`
- More details on the PVS specifications and example proofs of various BGP systems
- *Verifiable Policy-based Routing with DRIVER.*, Anduo Wang, Changbin Liu, Boon Thau Loo, Oleg Sokolsky, Prithwash Basu., University of Pennsylvania TR, MS-CIS-09-12, 2009.

Generating Equivalent NDlog implementation

- Specify atomic component t as a rule taking t.in(I) as rule body, deriving t.out(O) as rule head
  - PVS specification `t(I,O):INDUCTIVE bool = CT(I,O)`
  - The equivalent NDlog rule `t.out(O) :- t.in(I), CT(I,O)`
- Specify compositional component as set of rules

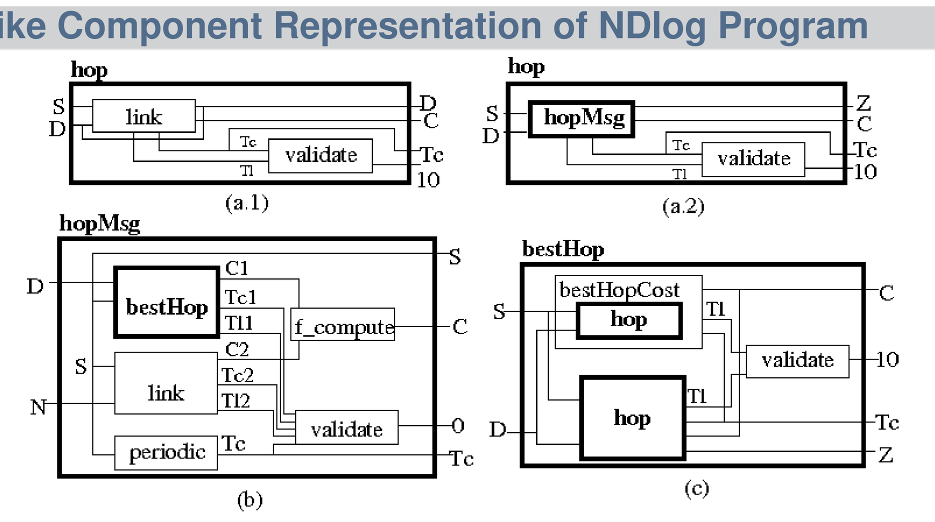
```
tc(I1,I2,O3): INDUCTIVE bool = EXISTS
(O1,O2):t1(I1,O1) AND t2(I2,O2) AND
t3(O1,O2,O3)

t1(I,O): INDUCTIVE bool = C1(I,O)
t2(I,O): INDUCTIVE bool = C2(I,O)
t3(I,O',O): INDUCTIVE bool = C3(I,I',O)
```



- Equivalent NDlog implementations
  - `t1.out(O1) :- t1.in(I1), C1(I1,O1).`
    `t2.out(O2) :- t2.in(I2), C2(I2,O2).`
    `t3.out(O3) :- t1.out(O1), t2.out(O2), C3(O1,O2,O3).`

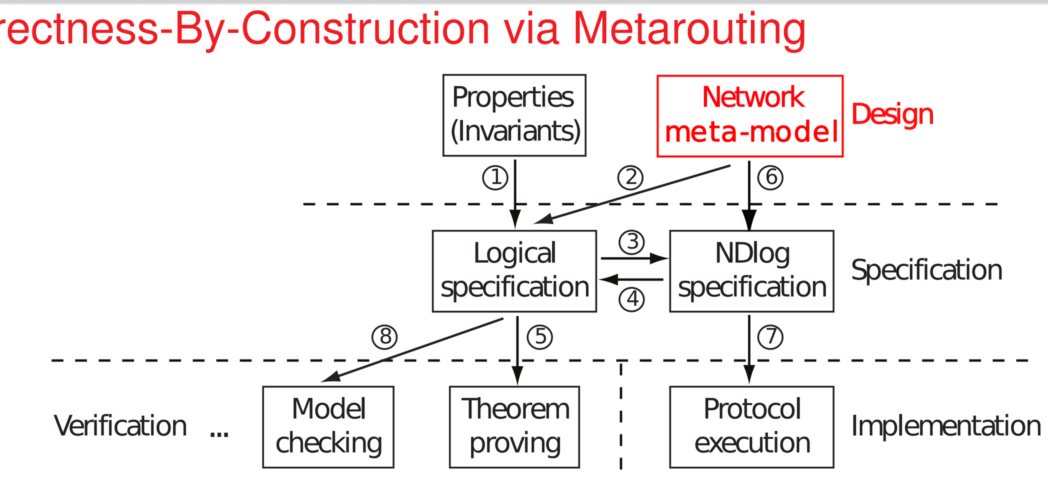
Circuit-Like Component Representation of NDlog Program




```
a1: hop(@S,D,D,C,Tc,10):-link(@S,D,C,Tc,10).
a2: hop(@S,D,Z,C,Tc,10):-hopMsg(@S,D,Z,C,Tc2),Tc=Tc2+5
agg bestHopCost(@S,D,min<C>,Tc,10):- hop(@S,D,D,C,Tc,10).
b: hopMsg(@N,D,Z,C,Tc,0):- periodic_dv(@S,5,Tc), link(@S,N,C2,Tc2,10),
bestHop(@S,D,Z,C1,Tc1,10),C=C1+C2,Tc2<Tc<=Tc2+10,Tc1<Tc<=Tc1+10.
c: bestHop(@S,D,Z,C,Tc,10):- bestHopCost(@S,D,C,Tc,10),
hop(@S,D,Z,C,Tc1,10), Tc1<Tc<=Tc1+10.
```

Meta-Theoretic Model

Correctness-By-Construction via Metarouting



- Metarouting, algebraic framework for routing protocol
  - Models BGP systems (today's de facto Internet routing) with convergence guarantee
- Our contribution: Formalize Metarouting theory in FVN using PVS
  - Heavy and interesting use of PVS theory interpretation: mapping and declaration
  - Extend PVS specification logic with metarouting theory
- Our goal: Free network operator from the tedious low-level and trivial theory consistency checking by leveraging PVS specification language and proof environment

Metarouting

Algebraic framework for modeling BGP systems

- Abstract routing algebra, mathematical model: $A = \langle \Sigma, \preceq, \mathcal{L}, \oplus, \mathcal{O}, \phi \rangle$

sorts Σ (paths), $\mathcal{L}$ (links)

opns $\preceq$: $\Sigma \times \Sigma \to$ *bool* (preference relation)

$\oplus : \mathcal{L} \times \Sigma \to \Sigma$ (label application function)

$\mathcal{O}$: *subset of* $\mathcal{L}$ (origination set)

$\phi : \Sigma$ (prohibited path)

axioms $\forall_{\alpha \in \Sigma - \{\phi\}} \quad \alpha \preceq \phi$ (*Maximality*)

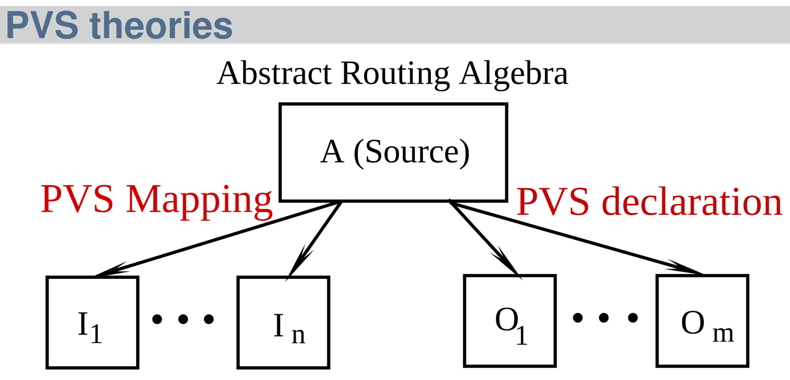
$\forall_{l \in \mathcal{L}} \quad l \oplus \phi = \phi$ (*Absorption*)

$\forall_{l \in \mathcal{L}} \forall_{\alpha \in \Sigma} \quad \alpha \preceq l \oplus \alpha$ (*Monotonicity*)

$\forall_{l \in \mathcal{L}} \forall_{\alpha, \beta \in \Sigma} \quad \alpha \preceq \beta \implies l \oplus \alpha \preceq l \oplus \beta$ (*Isotonicity*)

- Base algebras, atomic building blocks for shortest-path routing, customer-provider relationship etc
- Lexical product for route selection, composition operator that enables routing algebras composition
- *Isotonicity and Monotonicity* sufficient conditions for protocol convergence

Overview of PVS theories



- A: uninterpreted source theory `routeAlgebra`
- I_i: interpreted theory instantiated from A
- O_i: PVS theory taking routing algebra theories as parameters

Abstract Routing Algebra in PVS

```
routeAlgebra: THEORY
 BEGIN
  sig: TYPE+
  label: TYPE+

  injected: [label → bool]
  org: TYPE = {l: label | injected(l)}
  prohibitPath: sig
  labelApply: [label, sig → sig]
  prefRel: [sig, sig → bool]
  eqRel(s1, s2: sig): bool = prefRel(s1, s2) ∧ prefRel(s2, s1)
  mono(l: label, s: sig): bool = prefRel(s, labelApply(l, s))

  pref_complete: AXIOM
    ∀ (x, y: sig): prefRel(x, y) ∨ prefRel(y, x)
  absorption: AXIOM
    ∀ (l: label): labelApply(l, prohibitPath) = prohibitPath
  maximality: AXIOM ∀ (s: sig): prefRel(s, prohibitPath)
  monotonicity: AXIOM ∀ (l: label, s: sig): mono(l, s)
  isotonicity: AXIOM
    ∀ (s1, s2: sig)(l: label):
      prefRel(s1, s2) ⇒
        prefRel(labelApply(l, s1), labelApply(l, s2))
 END routeAlgebra
```

Base Algebra for Shortest Path Routing

Abstract Algebra `routeAlgebra` →

Base Algebra `addA`

- PVS mapping makes instantiations of uninterpreted types

```
sig ← upto(m+1)
label ← upto(n)
prohibitPath ← m+1
labelApply ← APPLY
prefRel ← PREF
```

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

Shortest Path Routing in PVS

- `routeAlgebra` Interpreted Theory: Base Algebra `addA`
- addA: THEORY

```
 BEGIN
  n: posnat
  m: posnat
  redundant: posnat
  N_M: AXIOM n < m
  LABEL: TYPE = upto(n)
  SIG: TYPE = upto(m+1)
  PREF(s1, s2: SIG): bool = (s1 ≤ s2)
  APPLY(l: LABEL, s: SIG): SIG =
     IF (l+s < m+1)
       THEN (l+s)
     ELSE (m+1)
     ENDIF
  IMPORTING routeAlgebra
         {{sig := SIG, label := LABEL, prohibitPath := m+1,
           labelApply(l: LABEL, s: SIG) := APPLY(l, s),
           prefRel(s1, s2: SIG) := (s1 ≤ s2)}}
 END addA
```

Base Algebra for Provider-Customer, Peer-Peer Guideline

- For economical reasons, ISP reduces use of provider routes, and maximizes availability of customer routes
- Σ(*path*): $C/R/P$ (customer/peer/provider path)
- $\mathcal{L}$(*link*): $c/r/p$ (customer/peer/provider link)
- $\oplus$ (label application):

$\oplus$	C	R	P
c	C	C	C
r	R	R	R
p	P	P	P

- $\preceq$ (preference relation): $C \preceq R$, $R \preceq P$, $C \preceq P$

Provider-Customer, Peer-Peer Guideline in PVS

- For simplicity, rename labels and signatures: $c \leftarrow 1, r \leftarrow 2, p \leftarrow 3$ and $C \leftarrow 1, R \leftarrow 2, P \leftarrow 3$
- lpA: THEORY

```
 BEGIN
  SIG: TYPE = upto(3)
  LABEL: TYPE = upto(3)
  IMPORTING routeAlgebra
         {{sig := SIG, label := LABEL,
           labelApply(l: LABEL, s: SIG) := l,
           prefRel(s1, s2: SIG) := (s1 ≤ s2),}}
 END lpA
```

A Concrete BGP system

- Route paths are measured in terms of customer-provider relationship and distance cost
  - 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=lexProduct$[A_2, B_2]$

Future Work

- Network Models and Implementation
  - Relaxed Algebraic models for wider range of protocols
  - Alternative component-based models: Click, Xorp
- Modeling Soft-state in Declarative Networking
  - Soft-state specified as NDlog tuples (predicates) in tables that will timeout, enables eventual consistency
  - Linear logic, semantic foundation for verification of NDlog programs with soft-state
- Combining Verification Techniques
  - Theorem proving, declarative networking specific decision procedures/proof strategies/tactics
  - Model checking declarative networking, transitions updating routing tables are specified in linear logic
  - Model checking small network instance and use theorem proving to scale

{anduo,liminjia,changbl,boonloo,oleg}@cis.upenn.edu pbasu@bbn.com

http://http://netdb.cis.upenn.edu/fvn/