Declarative Network Verification

Anduo Wang[1]
Prithwish Basu[2] Boon Thau Loo[1] Oleg Sokolsky[1]

[1]University of Pennsylvania

[2]BBN technologies

PADL 09

Motivation

- Challenges to today's Internet
  - Unwanted and harmful traffic
  - Complexity and fragility in Internet routing

Motivation

- Challenges to today's Internet
  - Unwanted and harmful traffic
  - Complexity and fragility in Internet routing
- Proliferation of "overlay networks":
  - Resiliency (RON, SOSR, Detour...)
  - Scalable Lookup (Chord, Pastry, Tapestry,...)
  - Mobility (i3, DHARMA, HIP)
  - Security (SOS, OverDoSe)
  - Content-distribution (Akamai, CoralCDN)
  - Multicast (Overcast, ESM)

Motivation

- Challenges to today's Internet
  - Unwanted and harmful traffic
  - Complexity and fragility in Internet routing
- Proliferation of "overlay networks":
  - Resiliency (RON, SOSR, Detour...)
  - Scalable Lookup (Chord, Pastry, Tapestry,...)
  - Mobility (i3, DHARMA, HIP)
  - Security (SOS, OverDoSe)
  - Content-distribution (Akamai, CoralCDN)
  - Multicast (Overcast, ESM)
- Clean-slate Internet Design
  - NSF FIND (Future Internet Design)
  - GENI (Global Environment for Network Innovation)

Motivation

- Challenges to today's Internet
  - Unwanted and harmful traffic
  - Complexity and fragility in Internet routing
- Proliferation of "overlay networks":
  - Resiliency (RON, SOSR, Detour...)
  - Scalable Lookup (Chord, Pastry, Tapestry,...)
  - Mobility (i3, DHARMA, HIP)
  - Security (SOS, OverDoSe)
  - Content-distribution (Akamai, CoralCDN)
  - Multicast (Overcast, ESM)
- Clean-slate Internet Design
  - NSF FIND (Future Internet Design)
  - GENI (Global Environment for Network Innovation)

Needed: Better software tools for deploying and analyzing new network protocols and architectures

Recent Efforts in Practical Network Verification

- Runtime verification
  - `Pip` `[NSDI'06]`
  - `DS3` `[NSDI'08]`
- Static analysis
  - `Metarouting` `[SIGCOMM'05]`
- Model checking
  - `MaceMC` `[NSDI'07]` `best paper`
  - `CMC` `[NSDI'04]`

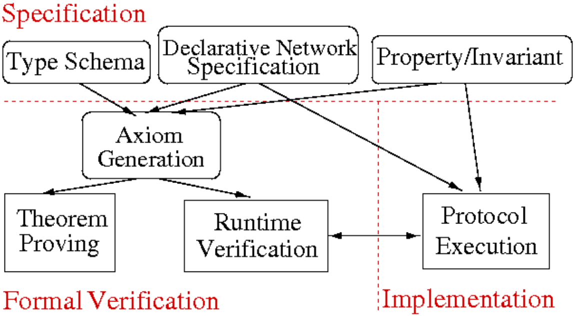
Limitations of Current Approaches

- Runtime verification
  - Incur additional runtime overhead
  - Non-exhaustive, limited class of properties
- Model checking network implementation
  - Require model extraction
  - State explosion problem:
    - Large state space persistent in network protocol prevents complete exploration
    - Restricted to temporal properties on small network

Limitations of Current Approaches

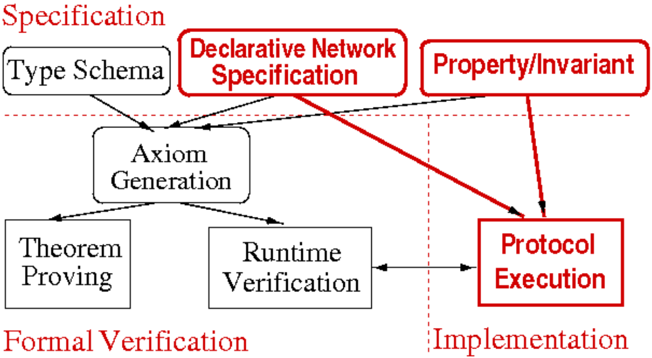
- Runtime verification
    - Incur additional runtime overhead
    - Non-exhaustive, limited class of properties
- Model checking network implementation
    - Require model extraction
    - State explosion problem:
        - Large state space persistent in network protocol prevents complete exploration
        - Restricted to temporal properties on small network
- Classical theorem proving
    - High initial investment in formal specification
    - Restricted to design and standard
    - Theorems are decoupled from actual implementation
    - Actual implementation not guaranteed to be error-free even when theorems are verified correct

Our Approach: DNV (Declarative Network Verification) bridges network specification, verification and implementation



1. Specification: *Declarative networking* code
2. Verification: General-purpose theorem prover
   - Automatic axiom generation process
3. Implementation: Distributed query processor

Background on Declarative Networking



See *Loo et. al* `[SOSP '05, SIGMOD '06]` for implementation details of declarative networking

Declarative Networking

A declarative framework for networks

- 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
- Advantages:
    - Ease of programming:
        - Compact high-level representation of protocols
        - Orders of magnitude reduction in code size

Declarative Networking

A declarative framework for networks

- 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
- Advantages:
    - Ease of programming:
        - Compact high-level representation of protocols
        - Orders of magnitude reduction in code size
    - Ease of analysis:
        - Amenable to static analysis and theorem proving

Network Datalog (`NDlog`) by example

All-Pairs Reachability

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

- *input*: `link(@S,D)`, *output*:`reachable(@S,D)`
- `link(@S,D)`:a link from node S to `D`, `reachable(@S,D)`: node `S` can reach `D`
- *Location specifier:* value of attribute prefixed with `@` determines the location of each tuple

Network Datalog (`NDlog`) by example

All-Pairs Reachability

-
  ```
  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` if there is a link from `S` to `D`
  - *input*: `link(@S,D)`, *output*:`reachable(@S,D)`
  - `link(@S,D)`:a link from node S to `D`, `reachable(@S,D)`: node `S` can reach `D`
  - *Location specifier:* value of attribute prefixed with `@` determines the location of each tuple

Network Datalog (`NDlog`) by example

All-Pairs Reachability

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

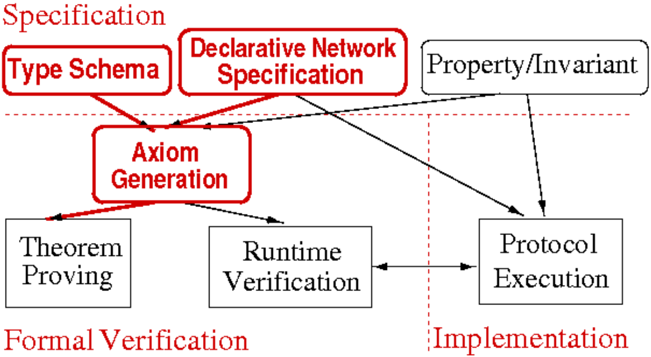
- For all nodes `S`,`D`,`Z`: if there is a link from `S` to `Z`, and that `Z` can reach `D`, then `S` can reach `D`

- *input*: `link(@S,D)`, *output*:`reachable(@S,D)`
- `link(@S,D)`:a link from node S to `D`, `reachable(@S,D)`: node `S` can reach `D`
- *Location specifier:* value of attribute prefixed with `@` determines the location of each tuple

Declarative Networking in Practice

- Example implementations to date:
  - Wired and wireless routing protocols (DV, LS, DSR, AODV, OLSR, etc.) `[SIGCOMM '05, PRESTO '08]`
  - Chord Distributed Hash Table `[SOSP '05]`
  - Resilient overlay network (RON) `[CoNEXT '08]`
  - Internet Indirection Infrastructure (i3) `[CoNEXT '08]`
  - Others: sensor networking protocols `[Sensys '07]`, multicast overlays, replication, snapshot, fault tolerance
- P2 declarative networking system
  - http://p2.cs.berkeley.edu

Automatic axiom generation process in DNV



PVS as our example theorem prover

Path Vector Routing in Network Datalog

```
p1 path(@S,D, , ):- link(@S,D, ),      .
p2 path(@S,D, , ):- link(@S,Z,  ),
path(@Z,D,  ,  ),        ,                   .
```

- Input: link(@source, destination,)
- Output: path(@source, destination, ,)

Path Vector Routing in Network Datalog

```
p1 path(@S,D,P, ):- link(@S,D, ),P=(S,D).
p2 path(@S,D,P, ):- link(@S,Z,  ),
path(@Z,D,P2,  ),        , P=concatPath(Z,P2).
```

- Input: link(@source, destination,   )
- Output: path(@source, destination, pathVector,   )

Path Vector Routing in Network Datalog

```
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).
```

- Input: link(@source, destination, cost)
- Output: path(@source, destination, pathVector, cost)

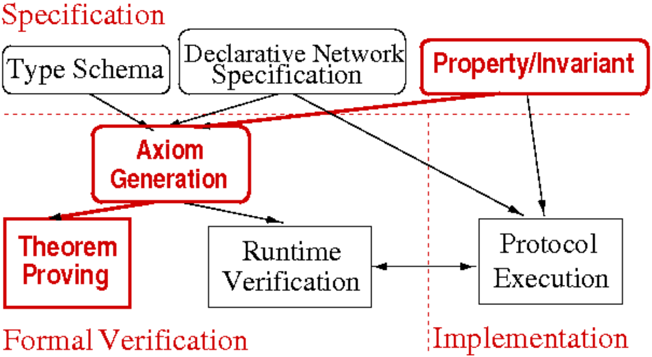
From Network Datalog to PVS Formalization

- Proof-theoretic semantics of path-vector routing
  - p1: $\forall(S,D,P,C).link(S,D,C) \wedge P = f_{init}(S,D) \implies path(S,D,P,C)$
  - p2: $\forall(S,D,P,C).\exists(C_1,C_2,Z,P_2).link(S,Z,C_1) \wedge bestPath(Z,D,P_2,C_2) \wedge C = C_1 + C_2 \wedge P = f_{concatPath}(Z,P_2) \implies path(S,D,P,C)$

- PVS equivalent formalization

```
path(S,D,(P: Path),C): INDUCTIVE bool =
 (link(S,D,C) AND P=f_init(S,D) AND Z=D) OR
 (EXISTS (C1,C2:Metric) (Z2:Node) (P2:Path):
   link(S,Z,C1) AND path(Z,D,P2,C2) AND C=C1+C2
   AND P=f_concatPath(S,P2) AND f_inPath(S,P2)=FALSE)
```

Verification using PVS



Example Verification using PVS

- Route optimality property: does path vector routing computes shortest paths between all nodes?

```
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)
```

- PVS proof scripts

```
("" (skosimp*) (expand bestPath) (prop)
    (expand bestPathCost) (prop) (skosimp*)
    (inst -2 C2!1) (grind))
```

- See extended technical report for general techniques: `http://repository.upenn.edu/cis_reports/890/`

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. See paper for details

Distance Vector Routing Protocol

- Distance vector routing:
    - NDLog specifications similar to path-vector routing except only next hop (instead of entire path) is traversed
- An instance of a soft-state NDLog program
    - Nodes periodically advertise to their neighbors their best known distances to other destinations
    - Nodes use these advertisements to select the best neighbor along the shortest path to destination
    - Advertisements timed-out unless refreshed

Example Properties Verified by DNV

Distance Vector Protocol with 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

Conclusion

- DNV: a unified framework that combines specification, verification, and implementation
  - Uses declarative networking with automatic axiom generation for theorem prover
- Ongoing and future work
  - More verification use cases
    - Variety of routing protocols (DSR, AODV, LS, etc.)
    - Policy-based Inter-domain routing
  - Integrating model checking into DNV
    - Semi-automatic model extraction from declarative network specification
    - Convergence and network invariants in temporal logic
  - Semi-automating the interactive proof process
    - Remove the theorem proving expert
    - Domain-specific proof strategies

Thank you

`http://www.seas.upenn.edu/ anduo/dnv.html`