Applying PL and Database Techniques to Networking

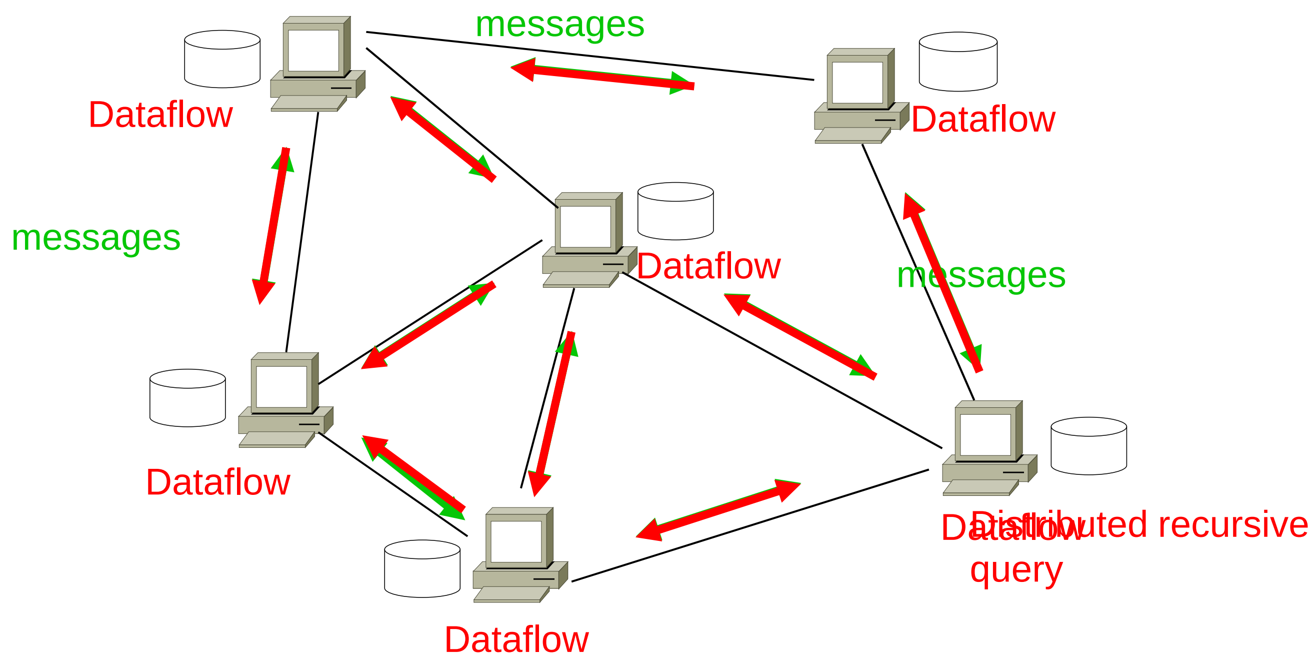
Boon Thau Loo
University of Pennsylvania
http://netdb.cis.upenn.edu

DIMACS Workshop on Designing Networks for Manageability (Nov 12-13)

Outline

- Declarative networking
  - Background
  - Uses in security, network monitoring, and formal verification
- Related work:
  - Flow-based Management Language (FML) – WREN'09
  - Functional languages in networking

Background: Declarative Network



Traditional Networks	Declarative Networks
Network State	Distributed database
Network protocol	Recursive Query Execution
Network messages	Distributed Dataflow

The Case for Declarative Networking

- **Ease of programming:**
  - Compact and high-level representation of protocols
  - Orders of magnitude reduction in code size
    - Declarative Chord DHT is 48 lines instead of 10,000.
  - Easy customization

- **Verifiability:**
  - Queries are “sandboxed” within query processor
  - Potential for static analysis of safety and use of more sophisticated verification techniques

- **What about efficiency?**
  - No fundamental overhead when executing standard routing protocols
  - Application of well-studied query optimizations
  - Note: Same question was asked of relational databases in the 70’s.

Background: Network Reachability

- **Declarative query language for network protocols**
  - Network Datalog (NDlog) – distributed Datalog **[SIGCOMM ‘05, SIGMOD ‘06]**
  - Compiled to distributed dataflows, executed by distributed query engine
  - *Location specifiers* (@ symbol) indicate the source/destination of messages

- **Example: Network Reachability**

r1: reachable(@S,D) :- link(@S,D)

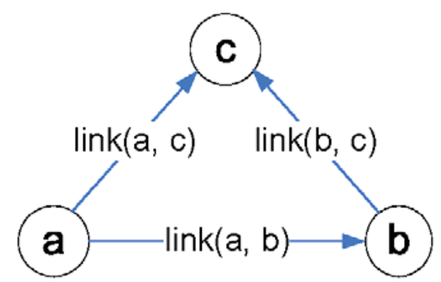
r2: reachable(@S,D) :- link(@S,Z), reachable(@Z,D)

link(@a,b) – “there is a link from node *a* to node *b*”

reachable(@a,b) – “node *a* can reach node *b*”

If there is a link from S to D, then S can reach D.

If there is a link from S to Z, AND Z can reach D, then S can reach D.



Node a	Node b
link(@a, b)	link(@b, c)
link(@a, c)	reachable(@b, c)
reachable(@a, c)	

Background: Path Vector

R1: path(@S,D,P) ← link(@S,D), P=(S,D).

R2: path(@S,D,P) ← link(@S,Z), path(@Z,D,P_2), P=S•P_2.

Query: path(@S,D,P)

Add S to front of P_2

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

Large Library of Declarative Protocols

- Example implementations to date:
  - **Routing protocols:** DV, LS, DSR, AODV, OLSR, HSLS, etc.
  - **Overlay networks:** Distributed Hash Tables, Resilient overlay network (RON), Internet Indirection Infrastructure (i3), P2P query processing, multicast trees/meshes, etc.
  - **Network composition:** Chord over RON, i3+RON **[CoNEXT'09]**
  - **Hybrid protocols**: Combining LS and HSLS, Epidemic + Proactive **[ICNP'09]**
  - **Others:** sensor networking protocols, replication, snapshot, fault tolerance protocols
- RapidNet (http://netdb.cis.upenn.edu/rapidnet/)
  - Declarative interface to the emerging ns-3 simulator [SIGCOMM'09 demo]
  - Open-source code release.
  - Deployment on ORBIT wireless testbed [WinTECH'09]

Outline

- Declarative networking
  - Background
  - Uses in **security**, network monitoring, and formal verification.
- Related work:
  - Flow-based Management Language (FML) – WREN'09
  - Functional languages in networking

Unified Declarative Platform for Secure Networked Information Systems.
Wenchao Zhou, Yun Mao, Boon Thau Loo, and Martín Abadi.
25th International Conference on Data Engineering (ICDE), 2009.

A3: An Extensible Platform for Application-Aware Anonymity.
Micah Sherr, Andrew Mao, William R. Marczak, Wenchao Zhou and Boon Thau Loo.
17th Annual Network & Distributed System Security Symposium (NDSS), 2010.

Formally Verifiable Networking.
Anduo Wang, Limin Jia, Changbin Liu, Boon Thau Loo, Oleg Sokolsky, and Prithwish Basu.
8th Workshop on Hot Topics in Networks (ACM SIGCOMM HotNets-VIII), 2009.

Access Control Logic

- **Access control languages:**
  - *Analyzing* and *implementing* security policies
  - Several runtime systems based on distributed Datalog/Prolog
- **Binder [Oakland 02]: a simple representative language**
  - **Context:** each principal has its own context where its rules and data reside
  - **Authentication:** "says" construct (digital signatures)

    At alice:

    b1: access(P,O,read) :- good(P).
    b2: access(P,O,read) :- bob says access(P,O,read).
  - "In alice's context, any principal P may access object O in read mode if P is good (b1) or, bob says P may do so (b2 - delegation)"
- **Several languages and systems:** KeyNote [RFC-2704], SD3 [Oakland 01], Delegation Logic [TISSEC 03], etc.

Unifying the two languages

- Declarative networking and access control languages are based on logic and Datalog
- Both extend Datalog in surprisingly similar ways
  - Notion of context (location) to identify components (nodes) in a distributed system
  - Suggests possibility to unify both languages
  - Leverage ideas from databases (e.g. efficient query processing and optimizations) to enforce access control policies
- Differences
  - Top-down vs bottom-up evaluation
  - Trust assumptions

Secure Network Datalog (SeNDlog)

- **Rules within a context**
  - Untrusted network
  - Predicates in rule body in local context
- **Authenticated communication**
  - "says" construct
  - *Export predicate:* "X says p@Y"
    - X exports the predicate p to Y.
  - *Import predicate:* "X says p"
    - X asserts the predicate p.

```
r1: reachable(@S,D) :- link(@S,D).
r2: reachable(@S,D) :- link(@S,Z),
        reachable(@Z,D).
```

⇩ *localization rewrite*

At S:

```
s1: reachable(@S,D) :- link(S,D).
s2: linkD(D,S)@D :- link(S,D).
s3: reachable(S,D)@Z :- linkD(S,Z),
        reachable(S,D).
```

⇩ *authenticated communication*

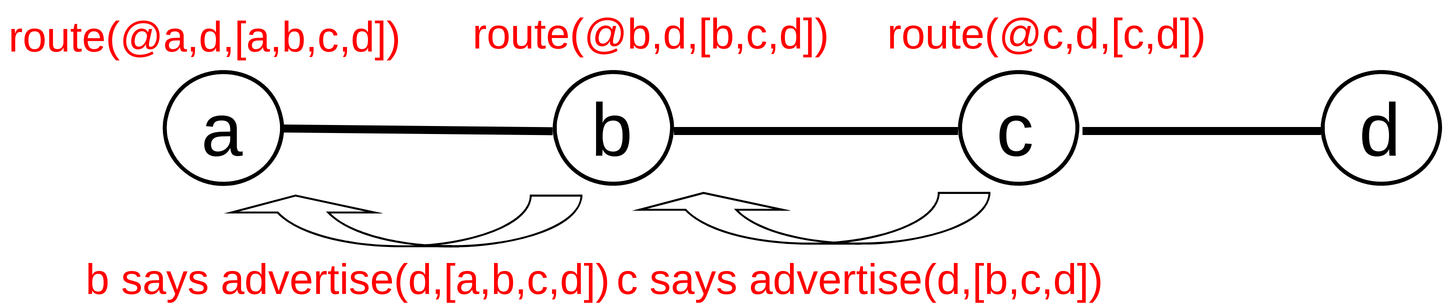
At S:

```
s1: reachable(S,D) :- link(S,D).
s2: S says linkD(D,S)@D :- link(S,D).
s3: S says reachable(S,D)@Z :-
        Z says linkD(S,Z),
        W says reachable(S,D).
```

Authenticated Path Vector Protocol

At Z,

z1 route(Z,X,P) :- neighbor(Z,X), P=f_initPath(Z,X).
z2 route(Z,Y,P) :- X says advertise(Y,P), acceptRoute(Z,X,Y).
z3 advertise(Y,P1)@X :- neighbor(Z,X), route(Z,Y,P),
carryTraffic(Z,X,Y), P1=f_concat(X,P).



Example Protocols in SeNDlog

- **Secure network routing**
  - Nodes import/export signed route advertisements from neighbors
  - Advertisements include signed sub-paths (*authenticated provenance*)
  - Building blocks for secure BGP
- **Secure packet forwarding**
- **Customizable anonymous routing [NDSS'10]**
  - Path selection and setting up "onion routes" with layered encryption
  - Application-aware Anonymity (http://a3.cis.upenn.edu)
- **Secure DHTs**
  - Chord DHT – authenticate the node-join process
  - Signed node identifiers to prevent malicious nodes from joining the DHT
- **Secure Distributed Query Processing**
  - PIER - built upon Chord DHT
  - Capability of *layered authentication*

Outline

- Declarative networking
  - Background
  - Uses in security, **network monitoring**, and formal verification.
- Related work:
  - Flow-based Management Language (FML) – WREN’09
  - Functional languages in networking

Runtime Network Monitoring via Queries

s1 eNewRoute(@S,D,P,T) :- bestPath(@S,D,P), T=f_now().

→ s2 persistence(@S,D,Diff) :- eNewRoute(@S,D,P,T), lastChange(@S,D,P1,T1),
P1 != P, Diff = T-T1.

→ s3 lastChange(@S,D,P,T) :- eNewRoute(@S,D,P,T).

→ s4 eAlarm(@S,D) :- persistence(@S,D,Diff), Diff<THRESHOLD.

eNewRoute(@S,D,P,T) – "a new route from S to D with path P is advertised at time T"
lastChange(@S,D,P,T) – "the route from S to D is last updated at time T"
persistence(@S,D,Diff) – "the route from S to D persists for Diff time"

Calculate *the route persistence* of the route from S to D

Record the last update time of the route

Raise an alarm when *the route persistence* is below a threshold

Distributed Event Correlation

```
materialize(pAlarms, keys(1,2), 5).
materialize(rAlarms, keys(1,2), 5).
materialize(gatewayAlarm, keys(1,2), 3600).

cm1 pAlarms(@M,S,D) :- persistenceAlarm(@S,D,Stat), gateway(@S,M).
cm2 rAlarms(@M,S,D) :- flowAlarm(@S,D,Stat), gateway(@S,M).
cm3 gatewayAlarm(@M,Z) :- pAlarms(@M,S,Z), rAlarms(@M,Z,D).
cm4 alarmNum(@M,COUNT<>) :- gatewayAlarm(@M,Z).
cm5 attackAlarm(@M,Count) :- alarmNum_ins(@M,Count), Count>4.
```

- *Soft-state* predicates with TTL allows time-based correlation.
- Analogous to time-based sliding window joins and aggregation in streaming databases (Borealis, Gigascope, TelegraphCQ)

Outline

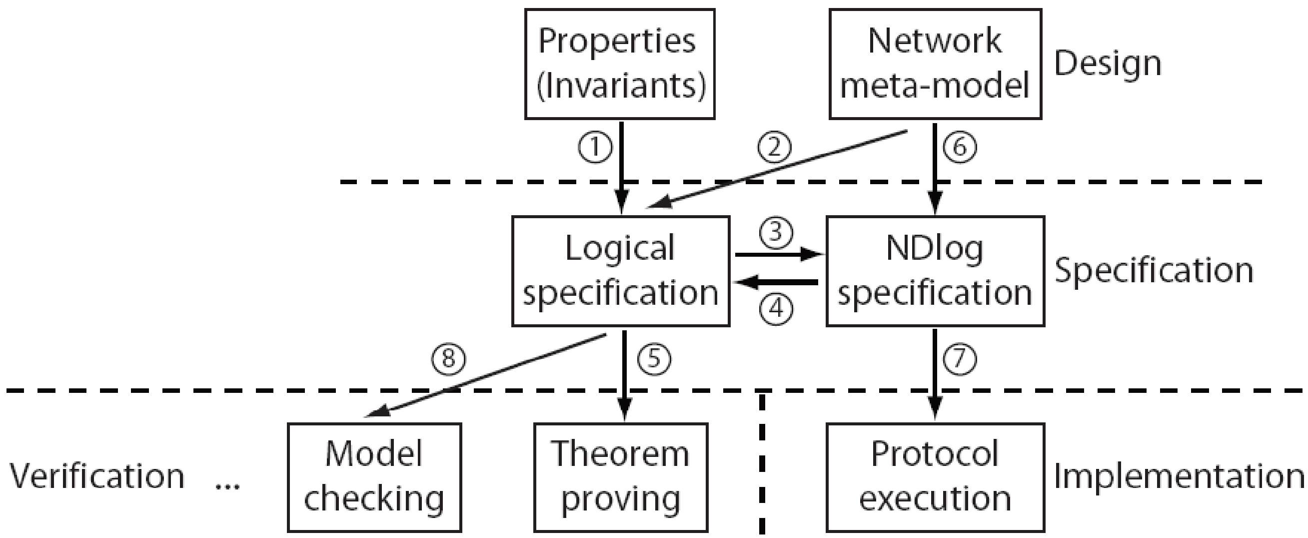
- Declarative networking
  - Background
  - Uses in security, network monitoring, and **formal verification**.
- Related work:
  - Flow-based Management Language (FML) – WREN'09
  - Functional languages in networking

Recent Efforts at Practical Network Verification

- **Runtime debugging**
  - Pip [NSDI'06], DS3 [NSDI'08]
  - Additional runtime overhead, inconclusive
- **Model checking**
  - CMC [NSDI'04], MaceMC [NSDI'07]
  - Inconclusive, restricted to temporal property and small networks
- **Theorem proving**
  - Initial high cost in specifying protocol design
  - Design decoupled from implementation
- **Correct-by-construction**
  - Metarouting [SIGCOMM'05]

Formally Verifiable Networking (FVN)

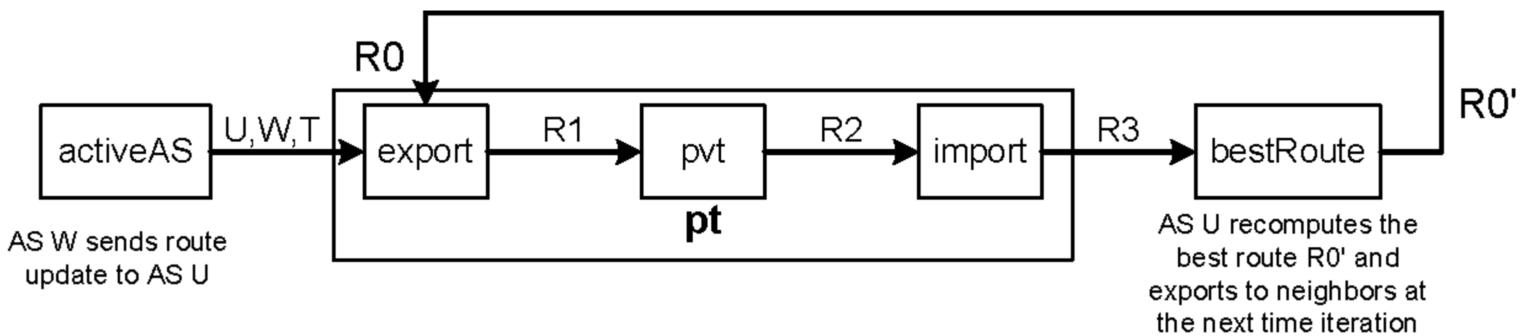
http://netdb.cis.upenn.edu/fvn



- Conceptually sound meta-model for program synthesis
  - Formal logical statements specify the behavior and the properties of the protocol
    - Declarative networking bridges logical specifications and actual implementation
- Theorem proving establishes correctness proof
  - System specification => property specification
  - Machine checked proof, proof automation support

Component-based Verification of BGP System

- Component model for BGP system based on route-transformation presented in: *An analysis of BGP convergence properties*, Timothy Griffin and Gordon Wilfong [SIGCOMM'99]



- Specification of BGP components in the PVS theorem prover
  - bgp(U,W,R0,R3,T): INDUCTIVE bool = activeAS(U,W,T) AND pt(U,W,R0,R3,T) AND bestRoute(W,T,R0)
  - pt(U,W,R0,R3,T): INDUCTIVE bool = EXISTS (R1,R2): export(U,W,R0,R1,T) AND pvt(U,W,R1,R2,T) AND import(U,W,R2,R3,T)

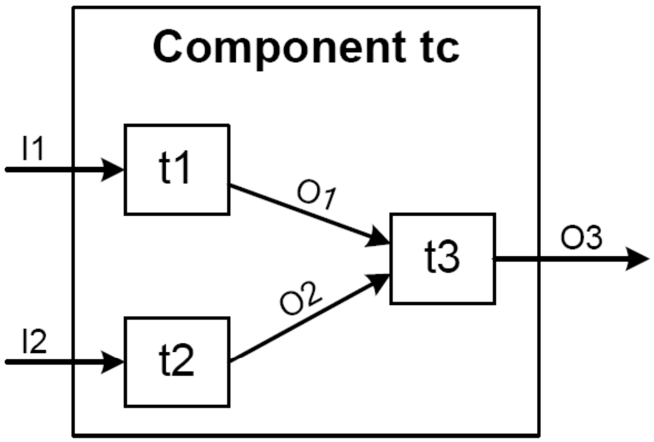
Verified NDlog Code Generation

- Given verified compositional component tc defined in terms of sub-components t1,t2,t3
- 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,I',O): INDUCTIVE bool = C3(I,I',O)



- Output from t1 and t2 are inputs to t3
- Equivalent NDlog rules:

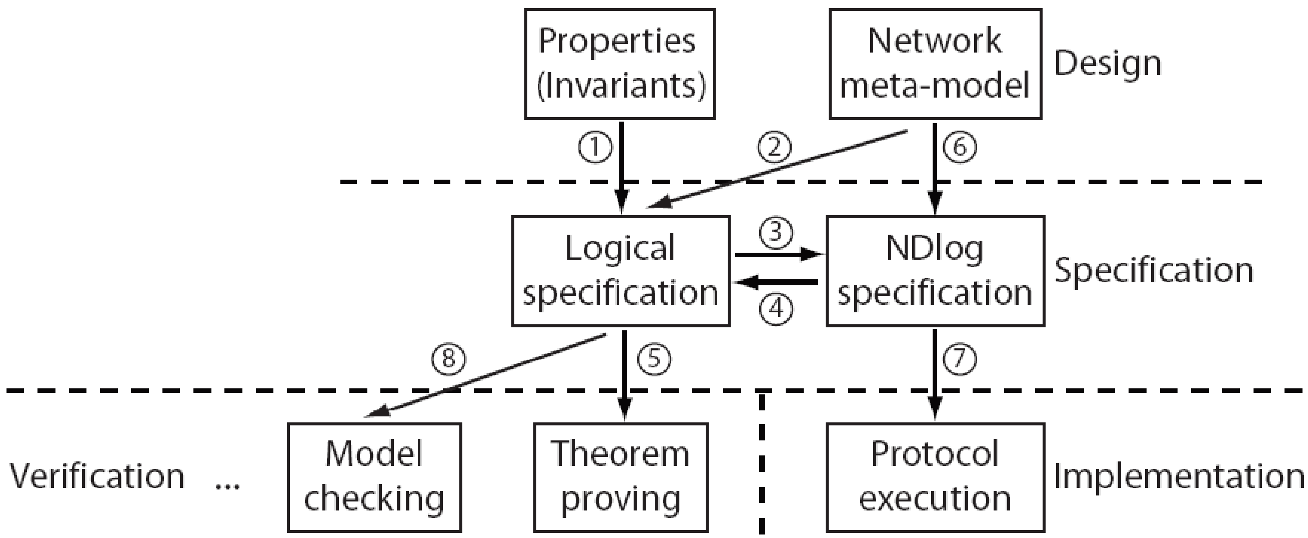
  t1 t1_out(O1) :- t1_in(I1), C1(I1,O1).

  t2 t2_out(O2) :- t2_in(I2), C2(I2,O2).

  t3 t3_out(O3) :- t1_out(O1), t2_out(O2), C3(O1,O2,O3).

Formally Verifiable Networking (FVN)

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



Declarative Network Verification. (arrows1, 4 & 5). [PADL'09]
Component-based Network Specifications. (arrows 2,3, & 5) [TPHOLs'09]
Formally Verifiable Networking (vision for all the other arrows) [HotNets'09]
Formalizing Metarouting in PVS [AFM'09]

Outline

- Declarative networking
  - Background
  - Uses in formal verification, network monitoring, security
- Related work:
  - Flow-based Management Language (FML) – WREN ‘09
  - Functional languages in networking

Practical Declarative Network Management.
Hinrichs, Gude, Casado, Mitchell, Shenker.
Workshop: Research on Enterprise Networking ’09.

Flow-based Management Language (FML)

- **Access control example**
  - `allow(<Flow>) :- superuser(Us).`
  - `superuser(todd).`
    `superuser(michelle).`
- **Access control keywords:** allow, deny, waypoint, avoid
- **<Flow> is a vector of 8 properties**

Property	Description
u_s and u_t	Source and Target Users
h_s and h_t	Source and Target Hosts
a_s and a_t	Source and Target Access Points
$prot$	Protocol
$request$	Whether or not flow is a request.

- **Another example**
  - `waypoint(<Flow>, ids) :- guest(Us), wireless(As).`
  - `wireless(wap1).`
  - `wireless(wap2).`

Outline

- Declarative networking
  - Background
- Related work:
  - Flow-based Management Language (FML) – WREN ‘09
  - Functional languages in networking

Functional Languages in Networking

- **What's functional programming?**
  - Treats computation as evaluation of mathematical functions.
  - Prominent languages: OCaML, Haskell, Erlang, Scheme

- **PLAN: Packet Language for Active Networks**
  - http://www.cis.upenn.edu/~dsl/PLAN/
  - Active networks: a programmable network infrastructure
  - PLAN uses a functional programming paradigm. Written in OCaML.
  - Limit the expressive power of language. Guaranteed to terminate.
  - Safety: Strongly-typed

Functional Languages in Networking

- **Nettle**
  - http://www.haskell.org/YaleHaskellGroupWiki/Nettle
  - Haskell-based configuration language for configuring BGP routers
  - Configurations are high-level, declarative, and platform implementation
  - Compile into router-specific configuration, e.g. XORP
- **Flask:**
  - http://www.eecs.harvard.edu/~mainland/flask/
  - Haskell-based programming environment for sensor networks
- **Opis:**
  - http://perso.eleves.bretagne.ens-cachan.fr/~dagand/opis/
  - OCaML based implementations of distributed systems
  - Higher-order and strongly typed.

Thank You …

http://netdb.cis.upenn.edu