Architectural Support for Internet Evolution and Innovation

George N. Rouskas

Department of Computer Science
North Carolina State University

http://net-silos.net/

Joint work with: Ilia Baldine (RENCI), Rudra Dutta, Anjing Wang, Mohan Iyer (NCSU)
Outline

- **Motivation:** Challenges with Internet Architecture
- **SILO:** A Meta-Design Framework
- **SILO as Research Tool:** Cross-Layer Experimentation
- Summary and Demo
Challenges with Current Architecture

1. **Evolution**: function-heavy protocols with built-in assumptions
2. **High barrier to entry**: for new data transfer protocols
3. **Cross-layer design**: lack of inter-layer interactions/controls
Several distinct functions:
- identify application endpoints (ports)
- e2e congestion control
- multi-homing (SCTP)
- reliability semantics (TCP, RDP, SCTP, etc)

→ evolution of individual functions affects entire transport layer

Lack of clear separation between policies and mechanisms
- window-based flow control vs. how window size may change
→ prevents reuse of various components

Built-in assumptions about IP addresses
→ transition to IPv6, support for mobility difficult
High Barrier to Entry

- New data transfer protocols difficult to implement/deploy
  - except for use-space
- Experimental network designs crucial for:
  - gaining insight
  - understanding protocol operation
  - discovering new knowledge rooted in physical world
- Implementations on commodity HW/SW remain challenging:
  - require modification of OS kernel
  - involve significant expertise
  - limit ability to “play” with network stack
Cross-layer design a major research theme over last decade:
- wireless networks
- TCP congestion control
- optical networks (later)
- ...

Adoption of ideas in operational networks quite slow:
- no interfaces for inter-layer interactions/cross-layer controls
- lack of experimental work
  → reliance on simulation with invalid assumptions
Accommodating New Functionality

- Deploy half-layer solutions (MPLS, IPSec)
  → layers become markers for vague functional boundaries
- Adapt existing implementation to new situations
  → TCP over wireless/large bw/delay product networks
- Implement own UDP-like data transfer
  → no reuse or kernel optimizations
- Abandon the old: new implementations for sensor networks
  → Internet balkanization
Role-Based Architecture (RBA) [BFH 2003]

- New abstraction: organize protocols in *heaps*, not stacks
- Richer interactions among protocols → flexibility
- Require new system-level implementations

![Diagram showing interactions and RSH 1, 2, 3]
Meta-protocol: generic protocol layer with basic services
Each layer in stack → appropriately configured instantiation
Allows reuse, cleaner cross-layer interactions, dynamic composition
**Recursive Network Architecture (RNA) [TP 2008]**

- **Meta-protocol**: generic protocol layer with basic services
- Each layer in stack → appropriately configured instantiation
- Allows reuse, cleaner cross-layer interactions, dynamic composition

![Diagram showing layers: Wireless, MP−1, MP−2, MP−3, MP−4]
Meta-protocol: generic protocol layer with basic services

Each layer in stack → appropriately configured instantiation

Allows reuse, cleaner cross-layer interactions, dynamic composition
Layering As Optimization Decomposition

- Protocol layers integrated into mathematical framework
  [CLCD 2007] [LSS 2006]
- Global optimization problem: network utility maximization
- Decomposition into subproblems → **layering**
  - optimal modules (protocols) map to different layers
  - interfaces between layers coordinate the subproblems
Layering As Optimization Decomposition

- Clean-state optimization → layered network architecture
  - optimal layering ≠ TCP/IP stack
  - various representations of optimization problem
    → different layered architectures
  - (loose) coupling among layers → cross-layer considerations
Our View

- Internet architecture houses an effective design
- **But:** it is not itself effective in enabling evolution
- New architecture must be designed for *adaptability/evolvability*
- New architecture must *preserve/generalize* layering
- SILO objective: *design for change*
What is Architecture?

- Fundamental elements/principles vs. design decisions
What is Architecture?

- Fundamental elements/principles vs. design decisions
- Diverse points of view → FIND projects target: addressing, naming, routing, protocol architecture, security, management, economics, communication technologies (wireless, optical), …
What is Architecture?

- Fundamental elements/principles vs. design decisions
- Diverse points of view → FIND projects target: addressing, naming, routing, protocol architecture, security, management, economics, communication technologies (wireless, optical), · · ·
- Our definition:
What is Architecture?

- Fundamental elements/principles vs. design decisions
- Diverse points of view → FIND projects target: addressing, naming, routing, protocol architecture, security, management, economics, communication technologies (wireless, optical), · · ·
- Our definition:

  it is precisely the characteristics of the system that does not change itself, but provides a framework within which the system design can change and evolve
Meta-Design Framework

- Obtain a meta-design that explicitly allows for future change
- Not a particular design or arrangement of specific features
Meta-Design Framework

- Obtain a meta-design that explicitly allows for future change
- Not a particular design or arrangement of specific features

The goal is not to design the “next” system, or the “best next” system, but rather a system that can sustain continuing change
**Building Blocks:** services of fine-grain functionality

**Design Principles:**

1. Generalize traditional layer stack
2. Enable inter-layer interactions:
   - knobs: explicit control interfaces
3. Design for change:
   - facilitate introduction of new services
4. Separate control from data functions
Generalization of Layering

- **Silo**: vertical composition of services
  - preserves layering principle
- **Per-flow** instantiation of silos
  - introduces flexibility and customization
- **Decoupling** of layers and services
  - services introduced at point in stack where necessary
**Knobs**: explicit control interfaces
- adjustable parameters specific to functionality of service
- enable info exchange among services

Algorithms may optimize jointly the behavior of services in a silo
Upward information passing
Inter-Layer Interactions (2)

Downward information passing
Inter-Layer Interactions (2)

Up-and-down information passing
Silo-wide optimization/calibration
Architecture does not dictate services to be implemented

Provide mechanisms to:

- introduce new services
- compose services into silos

Ontology of services: describes

- service semantics → function, data/control interfaces
- relationship among services → relative ordering constraints
Constraints on composing services A and B:
- A requires B
- A forbids B
- A must be above (below) B
- A must be immediately above (below) B
- Negations, AND, OR

Minimal set:
- Requires, Above, ImmAbove, NotImmAbove

All pairwise condition sets realizable
- Forbids = (A above B) AND (B above A)
- Above = NOT Below
Service Composition Problem

- Given: a set of essential services ← application
- Obtain a valid ordering of these and additional services
  - or, identify conflicts with constraints
- Simple composition algorithm implemented
- Ongoing research in formalizing the problem

Input

```
[Yellow]  [Light Blue]  [Brown]  [Light Blue]  [Blue]  [Purple]  [Teal]  [Green]
```

Output

```
[Yellow]  [Light Blue]  [Brown]  [Light Blue]  [Blue]  [Purple]  [Teal]  [Green]
```
The SILO Hourglass
http://net-silos.net/
Prototype Architecture

**Application**

```
SLIO API
```

**SILO Management Agent**

- Data
- Control

**SILO Tuning Agent**

- Optimization policies

**Tuning Strategies Storage**

- Method DSOs
- Knob descriptions

**SILO Construction Agent**

- Silo request/recipe
- Ontology access

**Universe of Services Storage**

- Packet traffic data and control channel
SILO As a Research Tool

- Deploys in a slice
- Researcher brings:
  - custom services
  - tuning algorithms
  - ontology updates
- Connect to measurement framework → cross-layer protocol experimentation tool
Optical substrate can no longer be viewed as black box
Software Defined Optics

- Optical substrate can no longer be viewed as black box
- Collection of intelligent and programmable resources:
Software Defined Optics

- Optical substrate can no longer be viewed as **black box**
- **Collection of** **intelligent and programmable** resources:
  - optical monitoring, sensing mechanisms
  - amplifiers, impairment compensation devices
  - tunable optical splitters
  - configurable add-drop
  - programmable mux-demux (e.g., adjust band size)
  - adjustable slot size
  - ...
Cross-Layer Interactions

- Impairment-aware RWA and network design
- Placement of optical sub-systems (converters, amplifiers, regenerators)
- Traffic grooming
- Inter-layer QoS and traffic engineering
- Optical layer multicast
- Multi-layer failure localization and recovery
- ...
Summary

- Vision – enable flexibility, evolution: “design for change”
  - fine-grain, reusable services, explicit control interface
  - enables experimentation, flexibility, community of innovation
  - per-flow service composition (silos)
  - ease of evolution, policies

- Framework – provide architectural support to vision:
  - constrained composition
  - commoditize cross-layer interaction / optimization
New research directions
- silos in the core and scalability
- policy enforcement through composition constraints
- (generalized) virtualization as a service

Extend the prototype
- portfolio of reusable services
- optical testbed deployment → breakable experimental net (BEN)

Explore synergies with other (FIND) projects