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
In Search of Next Generation Internet

Early Pioneer Work
- NewArch (DARPA)
- SIGCOMM FDNA

2000 01 02 03 04 05 06 07 08 09

- NSF FIND
- NSF GENI
- Pouzin Society
- Euro-NGI
- Euro-4WARD
- Euro-FIRE
- Asia Future Internet Forum
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
**Meta-protocol**: generic protocol layer with basic services

- Each layer in stack → appropriately configured instantiation
- Allows reuse, cleaner cross-layer interactions, dynamic composition

![Diagram](https://via.placeholder.com/150)
Recursive Network Architecture (RNA) [TP 2008]

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

Diagram:
- MP-4
- MP-3
- MP-2
- MP-1
- Wireless
- **Meta-protocol**: generic protocol layer with basic services
- Each layer in stack $\rightarrow$ 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 $\neq$ 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
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
Silos: Generalized Protocol Stacks

Cross-Service Tuning

- S1
- S4
- S5
- S7
- S8

Knobs

- S3
- S6
- S7
- S9

Silo & Service Mgmt

Composability Constraints

Physical Layers

App

Constraints

Phys. Layers

Architectural Support for Internet Evolution and Innovation

GaTech, September 23, 2009 – p.17
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
Downward information passing
Up-and-down information passing
Inter-Layer Interactions (2)

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 $\rightarrow$ function, data/control interfaces
- relationship among services $\rightarrow$ relative ordering constraints
Ontology – Networking Knowledge

Architectural Support for Internet Evolution and Innovation
Service Composition

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 $B) \text{ AND } (B$ above $A)$
- Above = NOT Below
Service Composition Problem

- Given: a set of essential services $\leftarrow$ 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
The SILO Hourglass
The SILO Hourglass

- Applications
- SILO Universe
- SILO
- Transport technologies
- SONET
- OTN
- PPP
- 802.11
- Ethernet
- 802.16
- Physical interfaces
class SILO_API {
public:
    SILO_API() {
        _silo_id = 0;
        _request_id_init = 0;
    }
    // build up a set of requires/forbids constraints
    // such as: APP requires/forbids services A,B
    ERROR_CODE create_required_srv (int request_id
    list<srv_ID, srv_list>);
    ERROR_CODE create_forbidden_srv (int request_id
    list<srv_ID, srv_list>);
    ERROR_CODE release_request (int request_id);
};

class SILO_Knob {
public:
    SILO_Knob();
    inline int get_max() { return _knob_intf_max; }
    inline int get_value() { return _knob_intf_value; }
    inline bool set_value(int mValue) {
        _knob_intf_value = mValue;
        return true;
    }
};

// register a silo, load the needed DBS
RecipId RegisterRecipe(const string &recipe);

// remove a recipe
void destroyRecipe(const RecipId &recipeId);

// create a new silo given a recipe ID
RecipId CreateSilo(const RecipId &recipeId);

// delete a silo
void destroySilo(const SiloId &siloid);

RecipId GetRecipeIdForSilo(const SiloId &siloid);

// manages a collection of silos and passes data through them
class SiloManager {
public:
    typedef unsigned int RecipId;
    typedef unsigned int SiloId;
    static SiloManager &Instance();

    // process data
    void ProcessData(const SiloId &siloid,
                     unsigned char *buf);
    void ProcessData(const SiloId &siloid,
                     unsigned char *buf);

http://net-silos.net/
Prototype Architecture

Application

SILO API

data
control

SILO Management Agent

SILO Tuning Agent

packet traffic
data and control channel

optimization policies

Tuning Strategies Storage

SILO Construction Agent

ontology access

Universe of Services Storage

silo request/recipe

method DSOs/knob descriptions
SILO As a Research Tool

- Deploys in a slice
- Researcher brings:
  - custom services
  - tuning algorithms
  - ontology updates
- Connect to measurement framework $\rightarrow$ 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
Ongoing Efforts

- 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