Net SILOs: Generalizing the Layered Network Architecture

George N. Rouskas

Department of Computer Science
North Carolina State University

http://net-silos.net/

Joint work with: Ilia Baldine (RENCI), Rudra Dutta (NCSU), Dan Stevenson (RTI), Anjing Wang (NCSU), Manoj Vellala (Cisco)
Outline

- Context: Layering and the Clean-Slate Debate
- Motivation: Software Defined Optics
- SILO Network Architecture: The Story So Far
- Summary and Future Directions
The Internet is broken!
The Internet is broken! (has ossified / reached an impasse)
The Internet is broken!

Security needs an overhaul: it is difficult to

- identify users
- hold them accountable for their actions
- prevent them from causing harm
The Internet is broken!

Middleboxes violate end-to-end principle:

- firewalls
- NAT
- proxies
The Internet is broken!

Fixed layer architecture is outdated
The Internet is broken!

Fixed layer architecture is outdated

Diagram:
- Physical
- Data Link
- Network
- Transport
- ssh
- App App App
The Internet is broken!

Fixed layer architecture is outdated
The Internet is broken!

Cross-layer interactions difficult: TCP over wireless
The Internet is broken!

Clear need for clean-state initiatives $\rightarrow$ NSF FIND, EU FIRE, \cdots

1. research in new network architectures
2. large-scale experimental facilities $\rightarrow$ GENI
The Internet is doing just fine, thank you!
The Internet is doing just fine, thank you!

- Biological metaphor: mutation and natural selection
- Evolutionary designs: more robust, less expensive
- Mid-layer protocols must be conserved – not ossified
  → innovation at lower/upper layers of architecture
The Internet is doing just fine, thank you!

- Biological metaphor: mutation and natural selection
- Evolutionary designs: more robust, less expensive
- Mid-layer protocols must be conserved – not ossified
  → innovation at lower/upper layers of architecture

→ Evolution beats revolution
Layering as optimization decomposition [CLCD 2007] [LSS 2006]

- Protocol layers integrated into mathematical framework
- Global optimization problem: network utility maximization
- Decomposition into subproblems $\rightarrow$ layering
  - Optimal modules (protocols) map to different layers
  - Interfaces between layers coordinate the subproblems
Layering as optimization decomposition [CLCD 2007] [LSS 2006]

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

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

- Internet architecture successful in accommodating change
Our View

- Internet architecture successful in accommodating change
- **But:** current practice of patches/tweaks cannot continue forever
Our View

- Internet architecture successful in accommodating change
- **But:** current practice of patches/tweaks cannot continue forever
- New architecture must be designed for adaptability/evolvability
Our View

- Internet architecture successful in accommodating change
- **But:** current practice of patches/tweaks cannot continue forever
- New architecture must be designed for **adaptability/evolvability**
- New architecture must **preserve/generalize** layering
Our View

- Internet architecture successful in accommodating change
- But: current practice of patches/tweaks cannot continue forever
- New architecture must be designed for adaptability/evolvability
- New architecture must preserve/generalize layering
- SILO objective:
Internet architecture successful in accommodating change

But: current practice of patches/tweaks cannot continue forever

New architecture must be designed for adaptability/evolvability

New architecture must preserve/generalize layering

SILO objective:

The goal is not to design the “next” system, or the “best next” system, but rather a system that can sustain continuing change
Software Defined Optics

- 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
- ...
SILO Architecture Highlights

- **Services**: building blocks of fine-grain functionality → reusability
- Generalizes traditional layer stack
- Enables inter-layer interactions:
  - **knobs**: explicit control interfaces
- Design for change:
  - facilitates introduction of new services
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

Knobs

Silo & Service Mgmt

Composability Constraints

Physical Layers
Inter-Layer Interactions (1)

- **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
Inter-Layer Interactions (2)

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 for:

- introduce new services
- compose services into silos

Ontology of services: describes

- service semantics → function, data/control interfaces
- relationship among services → relative ordering constraints
Ontology – Networking Knowledge

Net SILOs: Generalizing the Layered Network Architecture
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 $\leftarrow$ 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
SILO Software Prototype

```java
class SILO_API {
    public:
        SILO_API() {
            //siloid = 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_intMax; }
        inline int get_value() { return _knob_intValue; }
        inline bool set_value(int nValue) {
            _knob_intValue = nValue;
            return true;
        }
        // ...
};
```

// register a silo, load the needed DBs
RecipId RegisterRecipe(const string recipe);
// remove a recipe
void DestroyRecipe(const RecipId recipId);
// create a new silo given a recipe
const Siloid CreateSilo(const RecipId recipId);
// delete a silo
void DestroySilo(const Siloid siloid);
RecipId GetRecipeId(const Siloid siloid);

http://net-silos.net/
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
- software defined optics
- virtualization and slicing

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

Explore synergies with other (FIND) projects