The Future of Networking,
and the Past of Protocols

Scott Shenker
with Martín Casado, Teemu Koponen, Nick McKeown
(and many others....)
Software-Defined Networking

• SDN clearly has advantages over status quo

• But is SDN the “right” solution?

• My talk: Not what SDN is, but why SDN is
Key to Internet Success: Layers

Applications
...built on...
Reliable (or unreliable) transport
...built on...
Best-effort global packet delivery
...built on...
Best-effort local packet delivery
...built on...
Physical transfer of bits
The Importance of Layering

• Decomposed delivery into fundamental components

• Independent but compatible innovation at each layer

• An amazing success…

• …but an academic failure
Built an Artifact, Not a Discipline

• Other fields in “systems”: OS, DB, etc.
  - Teach basic principles
  - Are easily managed
  - Continue to evolve

• Networking:
  - Teach big bag of protocols
  - Notoriously difficult to manage
  - Evolves very slowly
Why Does Networking Lag Behind?

• Networks used to be simple

• New control requirements led to great complexity

• Fortunately, the infrastructure still works...
  - Only because of your great ability to master complexity

• This ability to master complexity is both a blessing...
  - ...and a curse!
A Simple Story About Complexity

• ~1985: Don Norman visits Xerox PARC
  - Talks about user interfaces and stick shifts
What Was His Point?

• The ability to *master complexity* is not the same as the ability to *extract simplicity*.

• When first getting systems to work…
  - Focus on mastering complexity

• When making system easy to use and understand
  - Focus on extracting simplicity

• You will never succeed in extracting simplicity
  - If don’t recognize it is different from mastering complexity
What Is *My* Point?

- Networking has never made the distinction…

- And therefore has never made the transition
  - Still focused on mastering complexity
  - Little emphasis on extracting simplicity from control plane

- Extracting simplicity builds intellectual foundations
  - Necessary for creating a discipline….
An Example Transition: Programming

- Machine languages: no abstractions
  - Mastering complexity was crucial

- Higher-level languages: OS and other abstractions
  - File system, virtual memory, abstract data types, ...

- Modern languages: even more abstractions
  - Object orientation, garbage collection, ...

Abstractions key to extracting simplicity
“The Power of Abstraction”

“Modularity based on abstraction is the way things get done”

− Barbara Liskov

Abstractions ➔ Interfaces ➔ Modularity

What abstractions do we have in networking?
Layers are Great Abstractions

- Layers only deal with the **data plane**
- We have no powerful **control plane** abstractions!
- How do we find those abstractions?
- Define our problem, and then decompose it.
The Network Control Problem

• Compute the configuration of each physical device
  - E.g., Forwarding tables, ACLs,…

• Operate without communication guarantees

• Operate within given network-level protocol

Only people who love complexity would find this a reasonable request
Programming Analogy

• What if programmers had to:
  - Specify where each bit was stored
  - Explicitly deal with all internal communication errors
  - Within a programming language with limited expressability

• Programmers would redefine problem:
  - Define a higher level abstraction for memory
  - Build on reliable communication abstractions
  - Use a more general language

• Abstractions divide problem into tractable pieces
  - Make task of control program easier…
From Requirements to Abstractions

1. Operate without communication guarantees
   Need an abstraction for **distributed state**

2. Compute the configuration of each physical device
   Need an abstraction that **simplifies configuration**

3. Operate within given network-level protocol
   Need an abstraction for general **forwarding model**
1. Distributed State Abstraction

- Shield mechanisms from vagaries of distributed state
  - While allowing access to this state

- Natural abstraction: *global network view*
  - Annotated network graph provided through an API

- Control mechanism is now program using API
  - No longer a distributed protocol, now just a graph algorithm
  - E.g. Use Dijkstra rather than Bellman-Ford
2. Specification Abstraction

• Control program should express desired behavior

• It should not be responsible for implementing that behavior on physical network infrastructure

• Natural abstraction: simplified model of network
  - Simple model only enough detail to specify goals

• This is “network virtualization”
Simple Example: Access Control

Abstract Network Model

Global Network View
3. Forwarding Abstraction

• Control plane needs flexible forwarding model

• Abstraction should not constrain control program
  - Should support whatever forwarding behaviors needed

• It should hide details of underlying hardware
  - Crucial for evolving beyond vendor-specific solutions
My Entire Talk in One Sentence

• SDN is defined precisely by these three abstractions
  - Distribution, forwarding, configuration

• SDN not just a random good idea…

• …can be “derived” from decomposing network control

• Fundamental validity and general applicability
Realizing These Abstractions

• Core challenge:
  - make distributed control problem a logically centralized one

• This involves designing a common distribution layer
  - Gathers information from network elements (topology)
  - Disseminates control commands to network elements

• This is done by a “Network Operating System”
Network of Defined (or Open) Routers

e.g. routing, access control

Control Program

Global Network View

Distributed algorithm running between neighbors

Network OS
Major Change in Paradigm

- No longer designing distributed control protocols
- Now just defining a centralized control function

Configuration = Function(view)

- This spells the end for distributed protocols
  - Easier to write, reason about, maintain, ….

- Beginning of the “software era” of networking
  - Rate of change, nature of standards, different culture, etc.
2. Specification Abstraction

• Abstraction is a simplified model of network

• Control program merely configures abstract model
  - Abstract configuration = Function(abstract model)

• Requires a new shared control layer:
  - Map abstract configuration to physical configuration
Software Defined Network

- Abstract Network Model
- Network Virtualization
- Global Network View
- Network OS
3. Forwarding Abstraction

- Switches have two “brains”
  - Management CPU (smart but slow)
  - Forwarding ASIC (fast but dumb)

- Need a forwarding abstraction for both
  - CPU abstraction can be almost anything

- ASIC abstraction is much more subtle: OpenFlow

- All this assumes switches are unit of abstraction
  - Why not fast, cheap fabrics? (in datacenters)
Keep in Mind

• As we debate the finer points of OpenFlow
  - OF is a detail in the overall SDN architecture

• OpenFlow is crucial for the industry
  - But a minor piece architecturally

• Don’t let the tail wag the dog…
Dog: Clean Separation of Concerns

• Control prgm: specify behavior on abstract model
  - Driven by Operator Requirements

• Net Virt’n: map abstract model to global view
  - Driven by Specification Abstraction

• NOS: map global view to physical switches
  - API: driven by Distributed State Abstraction
  - Switch/fabric interface: driven by Forwarding Abstraction
Final Words

• Future of networking lies in finding right abstractions
  - The era of “a new protocol per problem” is over

• Takes years to internalize and evaluate abstractions
  - Even longer to adjust to software-oriented culture

• We are in the early stages of an intellectual voyage
  - We should keep our minds open while charting our course
Thank You.....