01 Data center Fat tree

02 Data center Dcell

03 Data center PortLand

04 SDN 1 Ethane

05 SDN 2 OpenFlow

06 SDN 3 NOX

07 SDN 4 FlowVisor

08 SDN 5 P4

09 SDN 6 Protocol Oblivious Forwarding

10 SDN 7 Pyretic1

11 SDN 8 Pyretic2

12 Future Networking Named Content

13 Future DONA

14 Future Serval

**1.1 Fat-tree**

1. What is the datacenter network? What is the desired property of the datacenter network?

Data center is a pool of resources (computational, storage, network) interconnected using a communication network. Data Center Network (DCN) holds a pivotal role in a data center, as it interconnects all of the data center resources together. DCNs need to be scalable and efficient to connect tens or even hundreds of thousands of servers to handle the growing demands of Cloud computing. Today’s data centers are constrained by the interconnection network.

**Scalable interconnection bandwidth:** an arbitrary host in the data center can communicate with any other host in the network at the full bandwidth of its local network interface.

**Economies of scale:** make cheap off-the-shelf Ethernet switches the basis for large scale data center networks.

**Backward compatibility:** the entire system should be backward compatible with hosts running Ethernet and IP.

2. What is the traditional three-tier topology for the datacenter, its limitations?

A three-tiered design has a core tier in the root of the tree, an aggregation tier in the middle and an edge tier at the leaves of the tree.

**Oversubscription:** the ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology.

**Multi-path Routing:** Delivering full bandwidth between arbitrary hosts in larger clusters requires a “multi-rooted” tree with multiple core switches

**Cost:** 保证一定的oversubscription，cost会随规模急剧增加.

3. How Fat-tree differs from the traditional design? In Topology, Addressing, Routing paradigm

Topology: Fat tree DCN employs commodity network switches based architecture using Clos topology. The network elements in fat tree topology also follows hierarchical organization of network switches in access, aggregate, and core layers. However, the number of network switches is much larger than the three-tier DCN. The architecture is composed of k pods, where each pod contains, (k/2)2 servers, k/2 access layer switches, and k/2 aggregate layer switches in the topology. The core layers contain (k/2)2 core switches where each of the core switches is connected to one aggregate layer switch in each of the pods.

The fat tree architecture uses a customized addressing scheme and routing algorithm

Addressing:

All IP addresses in the network within the private 10.0.0.0/8 block.

The pod switches are given addresses of the form 10.pod.switch.1

Give core switches addresses of the form 10.k.j.i,

Routing paradigm:

Each entry in the main routing table will potentially have an additional pointer to a small secondary table of (suffix, port) entries.

**1.2 Dcell**

1. What is the physical structure of DCell?

 The DCell follows a recursively build hierarchy of cells. A cell0 is the basic unit and building block of DCell topology arranged in multiple levels, where a higher level cell contains multiple lower layer cells. The cell0 is building block of DCell topology, which contains *n* servers and one commodity network switch. The network switch is only used to connect the server within a cell0. A cell1 contain *k=n+1* cell0 cells, and similarly a cell2 contains k \* n + 1 cell1.

2. DCell properties:

Scalability: The number of servers scales doubly exponentially 

Fault-tolerance: The bisection width is larger than

3. How DCell route data flows? How to handle different types of failures?

Consider two nodes *src* and *dst* that are in the same *DCellk* but in two different

*DCellk*-1s. When computing the path from *src* to *dst* in a *DCellk*, we first calculate the intermediate link (*n*1, *n*2) that inter-connects the two *DCellk*-1s. Routing is then divided into how to find the two sub-paths from *src* to *n*1 and from *n*2 to dst. The final path of DCellRouting is the combination of the two sub-paths and (*n*1, *n*2).

DFR uses three techniques of local reroute, local link-state, and jump-up to address link failure, server failure, and rack failure, respectively.

**1.3 Portland**

1. Why existing L2 and L3 techniques have limitations in satisfying R1-5 for the cloud datacenter?

R1 and R2

Can be satisfied with a single layer 2 fabric, but layer 2 fabrics not scalable, need to support broadcasting.

A layer 3 fabric can not support VM migration .

R3

Layer 2: Require large number of MAC forwarding table entries, not feasible

R4 and R5

Layer 2 and 3 protocols (i.e. IS-IS, OSPF) are broadcast based, not efficient enough

2. How PortLand satisfies R1 – R5?

PortLand employs a logically centralized fabric manager that maintains soft state about network configuration information such as topology.

In PortLand, we restrict the amount of centralized knowledge and limit it to soft state. In this manner, we eliminate the need for any administrator configuration of the fabric manager (e.g., number of switches, their location, their identifier).

Replicated with a primary asynchronously updating state on one or more backups.

**2.1 Ethane**

1. Component of Ethane

A central Controller

Contains the global network policy and topology

Performs route computation for permitted flows.

A set of Ethane switches

Simple and dumb

Consisting of a simple flow table and a secure channel to the Controller

Forward packets under the direction of the Controller.

2. Name binding in a Ethane network. Why need name binding?

Ethane takes over all the binding of addresses, behave as a DHCP server

Machine is registered on the network

Users are required to authenticate with the network

The Controller can keep track of where any entity is located;

The Controller can journal all bindings and flow-entries in a log for network event reconstruction.

3. How two hosts communicate in a Ethane network?

Switch 1 forwards the packet to the Controller after determining that the packet does not match any active entries in its flow table.

The Controller decides whether to allow or deny the flow, or require it to traverse a set of waypoints.

The Controller computes the flow’s, adds a new entry to the flow tables of all the Switches along the path.

If path is allowed, the Controller sends the packet back to switch 1 which forwards it based on the new flow entry.

Subsequent packets from the flow are forwarded directly by the Switch, and are not sent to the Controller.

The flow-entry is kept in the switch until it times out.

4. Controller replicating

Three techniques for replicatin:

Cold standby: Backup Controllers sit idly-by waiting to take over if needed.

Warm-standby: a separate MST is created for every Controller.

Fully-replication: two or more active Controllers.

5. Policy Language

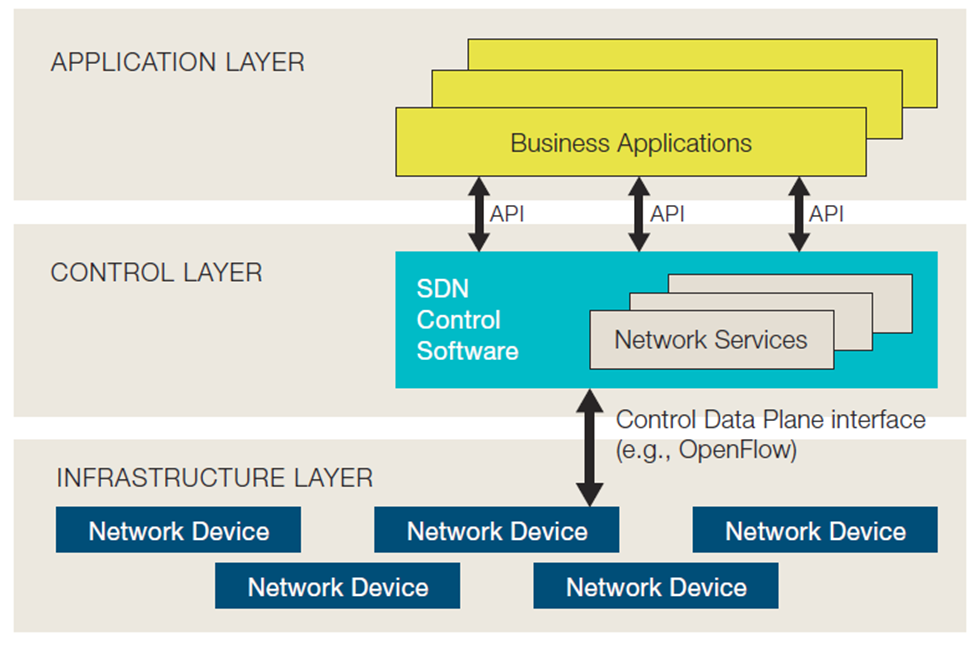
Ethane network policy is declared as a set of rules, each consisting of a condition and a corresponding action.  
 Rules are independent and don’t contain an intrinsic ordering.

**2.2 OpenFlow**

1. How SDN works?

decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.

2. Three layers in SDN? Where OpenFlow is located?



3. How to use SDN?

The traditional SDN approach, using a controller, northbound API, agents, and OpenFlow.

**2.3 NOX**

1. Why flow-based granularity is better than the packet-level and the prefix-level ones?

* 1. Choosing the granularity involves trading off between scalability vs. flexibility.
     1. A centralized per-packet control: infeasible to implement across any sizable network. No scalability.
     2. Prefix-based routing table: not allow sufficient control, since all packets between two hosts would have to follow the same path. No flexibility.
  2. An intermediate flow-based granularity
     1. Once control is exerted on some packet, subsequent packets with the same header are treated in the same way.

2. In NOX, which is the global network view, which is not?

NOX’s network view:

The switch-level topology; the locations of users, hosts, middle-boxes, and other network elements; and the services (e.g., HTTP or NFS) being offered.

All bindings between names and addresses.

Does not include the current state of network traffic.

**2.4 FlowVisor**

**2.5 P4**

1. What motivates P4. What is the key differences with OpenFlow 1.x.

Over the past five years, OpenFlow has grown increasingly more complicated

The proliferation of new header fields shows no signs of stopping.

Rather than repeatedly extending the OpenFlow specification, we argue that future switches should support flexible mechanisms for parsing packets and matching header fields, allowing controller applications to leverage these capabilities through a common, open interface

Such a general, extensible approach would be simpler, more elegant, and more future-proof than today's OpenFlow 1.x standard.

P4: used to congure a switch, telling it how packets are to be processed

OpenFlow1.x: that are designed to populate the forwarding tables in fixed function switches.

2. The components of the abstract forwarding model of a switch in P4.

A programming parser, Allow new headers to be defined

Multiple stages of match+action, In series, parallel, or combination of both

3. Table Dependency Graphs (TDG), TDG nodes map directly to match+action tables, and a dependency analysis identifies where each table may reside in the pipeline.

4. The P4 language: The major components.

Headers: describes the sequence and structure of a series of fields.

Parsers: specify how to identify headers and valid header sequences within packets.

Tables: Match+action tables are the mechanism for performing packet processing.

Actions: Construction of complex actions from simpler protocol-independent primitives.

Control Programs: determine the order of match+action tables that are applied to a packet.

5. How the language is compiled?

The compiler translates the parser description into a parsing state machine

Control program: not explicitly call out dependencies between tables or opportunities for concurrency. Employ a compiler to analyze the control program to identify dependencies and look for opportunities to process header fields in parallel. Finally, the compiler generates the target configuration for the switch.

**2.6 POF**

1. What are the challenges of OpenFlow?

Follows a reactive rather than proactive evolving path.

The forwarding plane is almost stateless. Lacks the capability to actively monitor flow status and change flow behavior without the involvement of the controller.

2. Why need a white box FE?

What the SDN really needs is a fully programmable forwarding plane in which the FEs are all white boxes.

3. Why POF is future proof?

Allow FEs to focus on performance rather than functionality

FE will become simpler and more flexible

**2.7 Pyretic1**

**2.8 Pyretic2**

1. Abstract packet model

1. Benefits
2. How to implement?

Each packet flowing through the network is a dictionary that maps field names to values.

Every field (including non-virtual ones) to hold a stack of values instead of a single bitstring.

Store in spare bits in the packet

2. What is network object?

Network object: allow programmers to abstract away details of the physical topology and write policies in terms of abstracted views of that network

1. Basic network object

The base network object represents the physical network.

1. Derived network object

A derived network object’s mapping comprises the following functions:

1. Mapping?

A function to map changes to the underlying topology up to changes on the derived topology,

A function to map policies written against the derived topology down to a semantically equivalent policy expressed only in terms of the underlying topology.

3. Policy transforming

Three auxiliary policies

1. Ingress\_policy: “lifts” packets in the underlying network up into the derived network by pushing appropriate switch and port identifiers onto the stack of values maintained in Pyretic’s abstract packet model.
2. Egress\_policy: “lowers” packets from the derived network to the underlying network by popping the switch and port identifier from the stack of values maintained in Pyretic’s abstract packet model.
3. Fabric\_policy: implements forwarding between adjacent ports in the derived network using the switches and links in the underlying topology.

**3.1 CCN**

1. Name some of evolutionary approaches for Internet development.

1. What is the major issue on evolutionary approach?

Availability: awkward, pre-planned, application-specific mechanisms are required. Example: P2P, CDN.

Security: Trust in content is easily misplaced, relying on untrustworthy location and connection information.

Location-dependence: Mapping content to host locations complicates configuration as well as implementation.

1. What is the other way for developing the Internet?

IPv6 IPSec Mobile IP DiffServ DHT

2. Three components of the CCN node, two types of packets in CCN.

CCN node has 3 components: FIB, Content Store and PIT

FIB: Forwarding table, allows multiple output faces

Content Store: Buffer, also caches Data packets

PIT: Pending Interest Table

Two packet types: Interest and Data

1. How users request contents?
2. How CCN node handles CCN packets?

Processing an Interest:

Matching Data is found in the Content Store   
=> send it and consume Interest

Pending Interest in PIT  
=> add this face to RequestingFaces list

Use FIB to forward Interest on outgoing faces, add to PIT

Processing Data:

Data follows a chain of PIT entries back to the source

Duplicate and unsolicited Data is discarded

3. How CCN name the content?

1. URI-like, hierarchical names

CCN is based on hierarchical, aggregatable names at least partly meaningful to humans

The name notation used is like URI

1. Names can be form a tree

The names form a tree which is traversed in preorder

In this way, the receiver can ask for the next Data packet in his Interest packet

**3.2 DONA**

1. How the host-centric networking causes the persistence, availability, and authentication issues.

Persistence: a data or service name remains valid as long as the data or service is available.

Availability: access to data and service should be reliable and have low-latency.

Authenticity: data came from the appropriate source, rather than from a adversary.

2. How DONA names the network entity?

Naming organized around principals

Names are of the form P:L

Granularity of naming is left up to principals

Names are “flat”

3. How a data is authenticated in DONA?

Self-certifying names

Principals are considered to own their data. A piece of data comes with the principal’s public key and the principal’s signature of the data.

Client receives the triplet <data, public key, signature>

If the client receive a piece of data with the name P:L, it can verify the data did come from the principal by

Checking the public key hashes into P

Validating that the signature corresponds to the public key

4. How name is resolved in DONA?

DONA uses the route-by-name paradigm for name resolution. Resolution infrastructure consists of Resolution handlers (RH).

Each domain will have one logical RH.

Name resolution is accomplished through the use of two basic primitives:

FIND(P:L) and REGISTER(P:L)

FIND(P:L) locate the object named P:L

REGISTER messages set up the state necessary for the RHs to route FINDs effectively

**3.3 Serval**

1. What are the features of the modern Internet service

* 1. Multiplicity
     1. Multiple replicas
     2. Multiple interfaces
     3. Multiple paths
  2. Dynamism
     1. Replica failure and recovery
     2. Service migration
     3. Client mobility
  3. Matches poorly with today’s TCP/IP stack

2. The Serval abstraction

Group-Based Service Naming

Service granularity

Format of serviceIDs

Securing communication and registration

Learning service names

Explicit Host-Local Flow Naming

Network-layer oblivious

Mobility and multiple paths

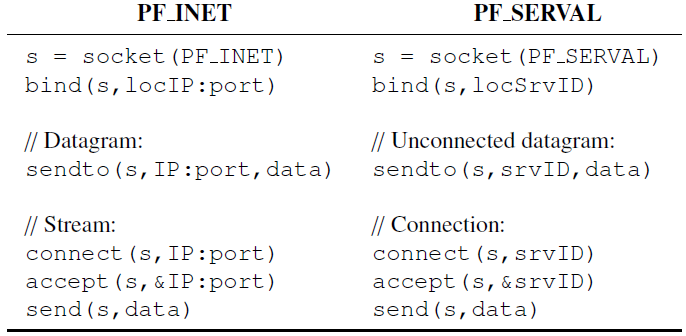
Middleboxes and NAT

No transport port numbers

Format and security

1. Application layer/Transport layer/Service access layer(SAL)/Network layer

3. Active socket vs. BSD socket



4. Network stack

1. Service table
2. Flow table
3. Service controller

the user-space service controller can manage service resolution based on policies, listen for servicerelated events, monitor service performance, and communicate with other controllers; the Service Access Layer (SAL) provides a service-level data plane responsible for connecting to services through forwarding over service tables. Once connected, the SAL maps the new flow to its socket in the flow table, ensuring incoming packets can be demultiplexed. Using in-band signaling, additional flows can be added to a connection and connectivity can be maintained across physical mobility and virtual migrations. Applications interact with the stack via active sockets that tie socket calls (e.g., bind and connect) directly to service-related events in the stack. These events cause updates to data-plane state and are also passed up to the control plane (which subsequently may use them toupdate resolution and registration systems).