第三章： Fat-Tree

Review

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

Today's data centers may contain tens of thousands of computers with significant aggregate bandwidth requirements. The network architecture typically consists of a tree of routing and switching elements with progressively more specialized and expensive equipment moving up the network hierarchy.

Desired Properties for a DC Network Architecture:

- 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?

Three tiers: core, aggregation, edge (ToR switch)

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.

Problems of the Topology

- Oversubscription: the ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology.
    - Ideal: 1:1: all hosts may potentially communicate with arbitrary other hosts at the full bandwidth of their network interface
    - Typical designs are oversubscribed by a factor of 2.5:1 (400 Mbps) to 8:1 (125 Mbps)
- Multi-path Routing: Delivering full bandwidth between arbitrary hosts in larger clusters requires a "multi-rooted" tree with multiple core switches
    - ECMP performs static load-balance, not a good choice （各路径的带宽、时延和可靠性等不一样，Cost＝hop count，不能很好地利用带宽，尤其在路径间差异大时，效果会非常不理想）
- Cost:
    - Using the largest 10 GigE and GigE switches to build a datacenter with 1:1 oversubscription
    - A cluster can be up to 27,648 hosts
- 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

- k pods, each containing two layers of $k/2$ switches.
- Each k-port switch in the lower layer is directly connected to $k/2$ hosts. Each of the

remaining $k/2$ ports is connected to $k/2$ of the k ports in the aggregation layer.

- $(k/2)^2$ k-port core switches. Each has one port connected to each of k pods. The i^{th} port of any core switch is connected to pod i such that consecutive ports in the aggregation layer of each pod switch are connected to core switches on $(k/2)$ strides.

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,
  - *pod* denotes the pod number (in [0, k−1]),
  - *switch* denotes the position of that switch in the pod (in [0, k−1], starting from left to right, bottom to top).
- Give core switches addresses of the form 10.*k.j.i*,
  - *j* and *i* denote that switch's coordinates in the $(k/2)^2$ core switch grid (each in [1, $(k/2)$], starting from top-left).

Routing paradigm

Two-level Routing Table

- Each entry in the main routing table will potentially have an additional pointer to a small secondary table of (*suffix*, *port*) entries.
  - A first-level prefix is terminating if it does not contain any second level suffixes,
  - A secondary table may be pointed to by more than one first-level prefix.

第 4 章： DCN-Dcell

What is the physical structure of DCell?

- $DCell_0$ is the building block to construct larger DCells. It has n servers and a mini-switch. All servers in $DCell_0$ are connected to the mini-switch.
  - n is a small integer (say, n=4).
- $DCell_1$ has n+1 =5 $DCell_0$s. $DCell_1$ connects the 5 $DCell_0$s as follows. Assign each server a 2-tuple [a_1, a_0], where a_1 and a_0 are the level-1 and level-0 IDs, respectively. Then two servers with 2-tuples [i, j-1] and [j, i] are connected with a link for every i and every $j > i$.
- For building $DCell_k$, if we have built $DCell_{k-1}$ and each $DCell_{k-1}$ has t_{k-1} servers, then we can create a maximum

  t_{k-1} + 1 of $DCell_{k-1}$ s.
- The number of $DCell_{k-1}$s in a $DCell_k$, g_k, and the total number of servers in a $DCell_k$ (i.e., t_k) are
  - $g_k=t_{k-1}+1$;
  - $t_k=g_k*t_{k-1}$

DCell properties:

- Scalability
- Fault-tolerance

How DCell route data flows?

How to handle different types of failures.

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

failure, and rack failure, respectively.

第 5 章： DCN-PortLand

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

How PortLand satisfies R1 – R5?

第 6 章： SDN-Ethane

Component of Ethane

- Controller
- Ethane switch

Name binding in a Ethane network

Why need name binding?

- Track all the bindings between names, addresses, and physical ports on the network event as Switches, hosts, and users join, leave, and move around the network.
- An Ethane Controller can journal all the authentication and binding information, it is possible to determine exactly which user sent a packet, when it was sent, the path it took, and its destination.

How two hosts communicate in a Ethane network?

UserA initiates a connection to userB (who we assume has already authenticated in a manner similar to userA). Switch

1 forwards the packet to the Controller after determining that

the packet does not match any active entries in its flow table.

2. On receipt of the packet, the Controller decides whether to allow or deny the flow, or require it to traverse a set of waypoints.

3. If the flow is allowed, the Controller computes the flow's route, including any policy-specified waypoints on the path. The Controller adds a new entry to the flow tables of all the Switches along the path.

- Controller replicating
- Policy Language

第 7 章： SDN-Openflow

How SDN works?

Three layers in SDN

Where OpenFlow is located?

- OpenFlow is a protocol that structures communication between the control and data planes under the context of software defined network.

How to user SDN?

第 8 章： SDN-NOX (Network operating systerm)

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

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

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

- Switch-level topology √
- Name binding √
- Traffic

第 10 章： P4

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

Motivation

- Over the past five years, OpenFlow has grown increasingly more complicated
- The proliferation of new header fields shows no signs of stopping.

differences

- 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 (i.e., a new “OpenFlow 2.0” API).
- Such a general, extensible approach would be simpler, more elegant, and more future-proof than today's OpenFlow 1.x standard.

P4: a higher-level language for Programming Protocol-independent Packet Processors

- Configure a switch, telling it how packets are to be processed
- Populate the forwarding tables in fixed function switches
- Compare with OpenFlow
    - Fixed parser
    - Series of actions

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

- A programming parser
- Multiple stages of match+action

TDG

- 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.

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.

How the language is compiled?

- Two-step compilation
    - At the highest level, programmers express packet processing programs using an imperative language representing the control flow (P4);
    - Below this, a compiler translates the P4 representation to TDGs to facilitate dependency analysis and then maps the TDG to a specific switch target.

第 11 章： SDN-POF

What are the challenges of OpenFlow?

- Challenges
    - 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.
- Consequences
    - The control plane and the forwarding plane are not sufficiently decoupled.
        - FE needs to have the knowledge of the protocol
    - There is no easy way to modify the packet format or add auxiliary data to the packets traveling in the network, let alone to apply user-defined protocols.
        - Innovation is in feasible with OpenFlow
        - Still need overlay, tunnel, etc.
- Consequences
    - Adding new forwarding protocol features requires an overhaul to both FE and controller, and in the worst case, a total redesign of the FE chip-set and hardware.
        - New protocols keep emerging, e.g., VXLAN, NVGRE
    - The limited expressivity can seriously impact the forwarding plane programmability.

Why need a white box FE?

- Current network devices are black boxes.

- OpenFlow and some other flavors of SDN proposed today make the network devices gray boxes with limited programmability.
    - The FEs maintain a fair amount of intelligence and the controller can do nothing beyond that.
- What the SDN really needs is a fully programmable forwarding plane in which the FEs are all white boxes.
- Control Plane can do whatever it want.
- The FEs should have a CPU-like role in the framework of SDN to ensure their flexibility and extensibility. That is, the FEs should not require any priori knowledge and understand the application semantics.

Why POF is future proof?

- Packet field parsing and handling are abstracted as generic instructions to enable flexible and future proof forwarding elements. This is simple yet has profound implications to SDN.

具体参见本章 ppt P10 P16 P17

第 14 章： Future-CCN

Name some of evolutionary approaches for Internet development.

- IPv6
- IPSec
- Mobile IP
- DiffServ
- DHT

What is the major issue on evolutionary approach?

- Issues:
    - **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.
        - Attack DNS.

What is the other way for developing the Internet?

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

How users request contents?

- Consumer broadcasts its *Interest* over all available connectivity
- *Data* is transmitted only in response to *Interest* and consumes that *Interest*

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

How CCN name the content?

- URI-like, hierarchical names
- Names can be form a tree

第 15 章： Future-DONA

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.

	Mechanism	**Issue(s)**
Persistence	**DNS, HTTP redirect**	**Neither work if data moving across domains**
Availability	**CDNs, P2P**	**Rely on application-specific and ad hoc mechanisms**
Authenticity	**IPsec, PKI,TLS**	**Focus on the channel, not on content**

How DONA names the network entity?

- Naming organized around principals

  Each principal is associated with a public-private key pair, and each datum or service or any other named entity is associated with a principal
- Names are of the form P:L

  P is the cryptographic hash of the principal's public key

  L is a label chosen by the principal, who ensures that these names are unique

How a data is authenticated in DONA?

- 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

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

第 16 章： Future-Serval

- **What are the features of the modern Internet service**
    - **Multiplicity**
    - **Dynamism**
- **The Serval abstraction**
    - **Application layer/Transport layer/Service access layer(SAL)/Network layer**
- **Active socket vs. BSD socket**
- **Network stack**
    - **Service table**
    - **Flow table**
    - **Service controller**