Scalable and Efficient Self-configuring Networks

Changhoon Kim
chkim@cs.princeton.edu
Today’s Networks Face New Challenges

• Networks **growing rapidly in size**
  – Up to tens of thousands to millions of hosts

• Networking environments getting **highly dynamic**
  – Mobility, traffic volatility, frequent re-adjustment

• Networks being increasingly deployed in **non-IT industries and developing regions**
  – Limited support for management

**Networks must be scalable and yet easy to build and manage!**
Pains In Large Dynamic Networks

• Control-plane overhead to store and disseminate host state
  – Millions of host-info entries
    ▪ Commodity switches/routers can store only ~32K/300K entries
  – Frequent state updates, each disseminated to thousands of switches/routers

• Status quo: Hierarchical addressing and routing

• Victimized ease of management
  – Leads to complex, high-maintenance, fragile, hard-to-debug, and brittle network
More Pains …

• **Limited data-plane capacity**
  – Tbps of traffic workload on core links
    ▪ Fastest links today are only 10 ~ 40 Gbps
  – Highly volatile traffic patterns

• **Status quo: Over-subscription**

• **Victimizes efficiency (performance)**
  – Lowers server utilization and end-to-end performance
  – Trying to mitigate this via traffic engineering can make management harder
All We Need Is Just A Huge L2 Switch, or An Abstraction of One!

- Host up to millions of hosts
- Obviate manual configuration
- Ensure high capacity and small end-to-end latency
Research Strategy and Goal

• Emphasis on architectural solutions
  – Redesign underlying networks

• Focus on edge networks
  – Enterprise, campus, and data-center networks
  – Virtual private networks (VPNs)

[ Research Goal ]
Design, build, and deploy architectural solutions enabling scalable, efficient, self-configuring edge networks
Why Is This Particularly Difficult?

• Scalability, self-configuration, and efficiency can conflict!
  – Self-configuration can significantly increase the amount of state and make traffic forwarding inflexible
  – Examples: Ethernet, VPN routing

Present solutions to resolve this impasse
Universal Hammers

- **Self-configuration**
  - **Flat Addressing**
    - Utilize location-independent names to identify hosts
    - Let network self-learn hosts’ info

- **Scalability**
  - **Traffic Indirection**
    - Deliver traffic through intermediaries (chosen systematically or randomly)

- **Efficiency**
  - **Usage-driven Optimization**
    - Populate routing and host state only when and where needed
Solutions For Various Edge Networks

**Enterprise, Campus Network**
- Config-free addressing and routing
- Huge overhead to store and disseminate individual hosts’ info
- Partition hosts’ info over switches

**Data-Center Network**
- Config-free addressing, routing, and traffic engineering
- Limited server-to-server capacity
- Spread traffic randomly over multiple paths

**Virtual Private Network**
- Config-free site-level addressing and routing
- Expensive router memory to store customers’ info
- Keep routing info only when it’s needed

**SEATTLE** [SIGCOMM’08]
**VL2** [SIGCOMM’09]
**Relaying** [SIGMETRICS’08]
Strategic Approach for Practical Solutions

Clean Slate

Enterprise, Campus Network
- Heterogeneity of end-host environments

Clean slate on network

Several independent prototypes

Data-Center Network
- Lack of programmability at switches and routers

Clean slate on end hosts

Real-world Deployment

Virtual Private Network
- Immediate deployment
- Transparency to customers

Backwards compatible

Pre-deployment Evaluation

Workable solution with sufficient improvement

SEATTLE VL2 Relaying

Backwards Compatible
SEATTLE: A Scalable Ethernet Architecture for Large Enterprises

Work with Matthew Caesar and Jennifer Rexford
Current Practice in Enterprises

A hybrid architecture comprised of several small Ethernet-based IP subnets interconnected by routers

- Loss of self-configuring capability
- Complexity in implementing policies
- Limited mobility support
- Inflexible route selection

Need a protocol that combines only the best of IP and Ethernet
## Objectives and Solutions

<table>
<thead>
<tr>
<th>Objective</th>
<th>Approach</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Avoid flooding</td>
<td>Resolve host info via unicast</td>
<td>Network-layer one-hop DHT</td>
</tr>
<tr>
<td>2. Restrain broadcasting</td>
<td>Bootstrap hosts via unicast</td>
<td>Traffic-driven resolution with caching</td>
</tr>
<tr>
<td>3. Reduce routing state</td>
<td>Populate host info only when and where needed</td>
<td>L2 link-state routing maintaining only switch-level topology</td>
</tr>
<tr>
<td>4. Enable shortest-path forwarding</td>
<td>Allow switches to learn topology</td>
<td></td>
</tr>
</tbody>
</table>

* Meanwhile, avoid modifying end hosts
Network-layer One-hop DHT

- Switches maintain `<key, value>` pairs by commonly using a hash function $F$
  - $F$: Consistent hash mapping a key to a switch
  - LS routing ensures each switch knows about all the other live switches, enabling one-hop DHT operations

- Unique benefits
  - Fast, efficient, and accurate reaction to churns
  - Reliability and capacity naturally growing with network size
Location Resolution

\(<\text{key, val}> = <\text{MAC addr, location}>\)

- Owner
  - Hash \(F(\text{MAC}_x) = B\)
  - Publish \(<\text{MAC}_x, A>\)

- Resolver
  - Store \(<\text{MAC}_x, A>\)

- Switches
- End hosts

Resources:
- Control message
- Data traffic
Handling Host Dynamics

• Host location, MAC-addr, or IP-addr can change

MAC- or IP-address change can be handled similarly
### Further Enhancements Implemented

<table>
<thead>
<tr>
<th>Goals</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handling network dynamics</td>
<td>Host-info re-publication and purging</td>
</tr>
<tr>
<td>Isolating host groups (e.g., VLAN)</td>
<td>Group-aware resolution</td>
</tr>
<tr>
<td>Supporting link-local broadcast</td>
<td>Per-group multicast</td>
</tr>
<tr>
<td>Dealing with switch-level heterogeneity</td>
<td>Virtual switches</td>
</tr>
<tr>
<td>Ensuring highly-available resolution service</td>
<td>Replicating host information</td>
</tr>
<tr>
<td>Dividing administrative control</td>
<td>Multi-level one-hop DHT</td>
</tr>
</tbody>
</table>
Prototype Implementation

• Link-state routing: **XORP OSPFD**
• Host-info management and traffic forwarding: **Click**
Amount of Routing State

SEATTLE reduces the amount of routing state by more than an order of magnitude

Table size (log)

Number of hosts ($h$)

- $y \approx 1.6h$
- $y \approx 2.4h$
- $y \approx 34.6h$
Cache Size vs. Stretch

Stretch = actual path length / shortest path length (in latency)

SEATTLE offers near-optimal stretch with very small amount of routing state
SEATTLE Conclusion

• Enterprises need a huge L2 switch
  – Config-free addressing and routing, support for mobility, and efficient use of links

• Key lessons
  – Coupling DHT with LS routing offers huge benefits
  – Reactive resolution and caching ensures scalability

Flat Addressing  MAC-address-based routing
Traffic Indirection  Forwarding through resolvers
Usage-driven Opt.  Ingress caching, reactive cache update
Further Questions

• What other kinds of networks need SEATTLE?
• What about other configuration tasks?
• What if we were allowed to modify hosts?

These motivate my next work for data centers
VL2: A Scalable and Flexible Data-Center Network

Work with Albert Greenberg, Navendu Jain, Srikanth Kandula, Dave Maltz, Parveen Patel, and Sudipta Sengupta
Data Centers

- Increasingly used for non-customer-facing decision-support computations
- Many of them will soon be outsourced to cloud-service providers
- Demand for large-scale, high-performance, cost-efficient DCs growing very fast
Tenets of Cloud-Service Data Center

• **Scaling out**: Use large pools of commodity resources
  – Achieves reliability, performance, low cost

• **Agility**: Assign any resources to any services
  – Increases efficiency (statistical multiplexing gain)

• **Low Management Complexity**: Self-configuring
  – Reduces operational expenses, avoids errors

Conventional DC network ensures none
Status Quo: Conventional DC Network

Internet

DC-Layer 3

DC-Layer 2

~ 1,000 servers/pod == IP subnet

Reference – “Data Center: Load balancing Data Center Services”, Cisco 2004

Key
- CR = Core Router (L3)
- AR = Access Router (L3)
- S = Switch (L2)
- LB = Load Balancer
- A = Rack of app. servers

Poor utilization and reliability
Status Quo: Traffic Patterns in a Cloud

• Instrumented a cluster used for data mining, then computed representative traffic matrices

• Traffic patterns are highly divergent
  – A large number (100+) of representative TMs needed to cover a day’s traffic

• Traffic patterns are unpredictable
  – No representative TM accounts for more than a few hundred seconds of traffic patterns

Optimization approaches might cause more trouble than benefits
## Objectives and Solutions

<table>
<thead>
<tr>
<th>Objective</th>
<th>Approach</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ensure layer-2 semantics</td>
<td>Employ flat addressing</td>
<td>Name-location separation &amp; resolution service</td>
</tr>
<tr>
<td>2. Offer uniform high capacity between servers</td>
<td>Guarantee bandwidth for hose-model traffic</td>
<td>Random traffic indirection (VLB) over a topology with huge aggregate capacity</td>
</tr>
<tr>
<td>3. Avoid hot spots w/o frequent reconfiguration</td>
<td>Use randomization to cope with volatility</td>
<td></td>
</tr>
</tbody>
</table>

* Embrace end systems!
Addressing and Routing: Name-Location Separation

Cope with host churns with very little overhead

- No LSA flooding for host info
- No broadcast msgs to update entire hosts
- No host or switch reconfiguration

Hosts use flat names

Resolution Service

...  
\text{x} \rightarrow \text{ToR}_2  
\text{y} \rightarrow \text{ToR}_3  
\text{z} \rightarrow \text{ToR}_3  
...

Lookup & Response
Example Topology: Clos Net

Offer huge **aggregate** capacity at modest cost
Traffic Forwarding: Random Indirection

Cope with arbitrary TMs with very little overhead

[ IP anycast + flow-based ECMP ]

- Harness huge bisection bandwidth
- Obviate esoteric traffic engineering or optimization
- Ensure robustness to failures
- Work with switch mechanisms available today

Links used for up paths
Links used for down paths
Implementation

**Switches**
- Commodity Ethernet ASICs
- Custom settings for line-rate decapsulation
- Default buffer-size settings
- No QoS or flow control

**Directory service**
- Replicated state-machine (RSM) servers, and lookup proxies
- Various distributed-systems techniques

**App servers**
- Custom Windows kernel for encapsulation & directory-service access
Data-Plane Evaluation

- **Ensures uniform high capacity**
  - Offered various TCP traffic patterns, then measured overall and per-flow goodput

<table>
<thead>
<tr>
<th></th>
<th>VL2</th>
<th>Fat Tree</th>
<th>Dcell</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goodput efficiency</strong></td>
<td>93+%</td>
<td>75+% (w/o opt)</td>
<td>40 ~ 60%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>95+% (with opt)</td>
<td></td>
</tr>
<tr>
<td><strong>Fairness between flows</strong></td>
<td>0.995(^\S)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

\(^\S\) Jain's fairness index defined as \(\frac{\left(\sum x_i\right)^2}{n \cdot \sum x_i^2}\)

- **Works nicely with real traffic as well**

![Graph showing Fairness Index over Time](image)
VL2 Conclusion

- Cloud-service DC needs a **huge L2 switch**
  - Uniform high capacity, oblivious TE, L2 semantics

- Key lessons
  - Hard to outsmart haphazardness; tame it with dice
  - Recipe for success: Intelligent hosts + Rock-solid network built with proven technologies

**Flat Addressing**
- Name-location separation

**Traffic Indirection**
- Random traffic spreading (VLB + ECMP)

**Usage-driven Opt.**
- Utilizing ARP, reactive cache update
Relaying: A Scalable VPN Routing Architecture

Work with Alex Gerber, Shubho Sen, Dan Pei, and Carsten Lund
Virtual Private Network

- Logically-isolated communication channels for corporate customers, overlayed over provider backbone
  - Direct any-to-any reachability among sites
  - Customers can avoid full-meshing via outsourcing routing
VPN Routing and Its Consequence

Site-level Flat Addressing:
Virtual PEs (VPEs) self-learn and maintain full routing state in the VPN (i.e., routes to every address block used in each site)

Memory footprint of a VPE (forwarding table size)
Mismatch in Usage of Router Resources

- Memory is full, whereas lots of ports still unused
- Revenue is proportional to provisioned bandwidth
- Large VPN with a thin connection per site is the worst case
- Unfortunately, there are many such worst cases
- Providers are seriously pinched
Key Questions

• What can we do better with existing resources and capabilities only, while maintaining complete transparency to customers?

• Do we really need to provide direct reachability for every pair of sites?
  
  – Even when most (84%) PEs communicate only with a small number (~10%) or popular PEs ...
Relaying Saves Memory

- Each VPN has two different types of PEs
  - **Hubs**: Keep full routing state of a VPN
  - **Spokes**: Keep local routes and a single default route to a hub
- A spoke uses a hub consistently for all non-local traffic
Real-World Deployment Requires A Mgmt-Support Tool

• Two operational problems to solve
  – Hub selection: Which PEs should be hubs?
  – Hub assignment: Which hub should a given spoke use?

• Constraint
  – Stretch penalty must be bounded to keep SLAs

• Solve the problems individually for each VPN
  – Hub selection and assignment decision for a VPN is totally independent of that of other VPNs
  – Ensures both simplicity and flexibility
Latency-Constrained Relaying (LCR)

• Notation
  – PE set: \( P = \{1, 2, \ldots, n\} \)
  – Hub set: \( H \subseteq P \)
  – The hub of PE \( i \): \( \text{hub}(i) \in H \)
  – Usage-based conversation matrix: \( C = (c_{i,j}) \)
  – Latency matrix: \( L = (l_{i,j}) \)

• Formulation
  – Choose \textbf{as few hubs as possible, while limiting additional distance due to Relaying}

\[
\begin{align*}
\min & \quad |H| \\
\text{s.t.} & \quad \forall s, d \in P \text{ whose } c_{s,d} = 1, \\
& \quad l_{s,\text{hub}(s)} + l_{\text{hub}(s),d} - l_{s,d} \leq \theta
\end{align*}
\]
Memory Saving and Cost of LCR

Based on entire traffic in 100+ VPNs for a week in May, 2007

LCR can save ~90% memory with very small path inflation

- Gain
- Cost

Fraction of routes removed
Fraction of traffic relayed
Increase of backbone load

<table>
<thead>
<tr>
<th>Maximum additional distance, theta (in miles)</th>
<th>Cost</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 msec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>~ 2.5 msec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>~ 11.5 msec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Deployment and Operation

• **Oblivious optimization** also leads to significant benefits

• Can implement this via minor routing protocol **configuration change** at PEs

• Performance degrades **very little** over time
  – Cost curves are fairly robust
  – Weekly/monthly adjustment: 94/91% of hubs remain as hubs

• Can ensure **high availability**
  – Need more than one hub located at different cities
  – 98.3% of VPNs spanning 10+ PEs have at least 2 hubs anyway
  – Enforcing “|H| > 1” reduces memory saving by only 0.5%
Relaying Conclusion

• VPN providers need a huge L2-like switch
  – Site-level PNP networking, any-to-any reachability, and scalability

• Key lessons
  – Traffic locality is our good friend
  – Presenting an immediately-deployable solution requires more than just designing an architecture

<table>
<thead>
<tr>
<th>Flat Addressing</th>
<th>Hierarchy-free site addressing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Indirection</td>
<td>Forwarding through a hub</td>
</tr>
<tr>
<td>Usage-driven Opt.</td>
<td>Popularity-driven hub selection</td>
</tr>
</tbody>
</table>
**Goals Attained**

- **Self-config**
  - SEATTLE
    - Eliminates addressing and routing configuration
    - Significantly reduces control overhead and memory consumption
    - Improves link utilization and convergence
  - VL2
    - Additionally eliminates configuration for traffic engineering
    - Allows a DC to host over 100K servers w/o oversubscription
    - Boosts DC-server utilization by enabling agility
  - Relaying
    - Retains self-configuring semantics for VPN customers
    - Allows existing routers to serve an order of mag. more VPNs
    - Only slightly increases end-to-end latency and traffic workload

- **Scalability**
- **Efficiency**
Summary and Future Work

• Designed, built, and deployed huge L2-like switches for various networks

• The universal hammers are applicable for various situations

• Future work
  – Other universal hammers?
  – Self-configuration on the Internet scale?
  – Architecture for distributed mini data centers?