SPARTA: Scalable Per-Address RoutIng Architecture

John Carter
Data Center Networking
IBM Research - Austin
IBM Research activities related to SDN / OpenFlow

IBM Research started a strategic initiative in data center networking in 2010
- Global participation from multiple labs, partnered with product teams
- SDN is one of the focus areas of the strategic initiative
- Heavily involved in ONF standards work (esp. FAWG → Table Typing Pattern)

### SDN applications (examples)

- Cloud network services
- Flow replication / recovery
- Security integration

### SDN advanced controller capabilities

- Network control apps, IT-network integration
- Application APIs, network abstractions
- Orchestration, workflows, network services
- Network device control and management (plugins / drivers)

### Network fabric / virtualization

- Scalable, flexible, converged data center fabric
- DOVE – distributed overlay virtual Ethernet

**OpenFlow mgmt tools**
Current SDN uses a tiny fraction of switch capabilities

- Previously proposed SDN routing architectures:
  - Largely based on OpenFlow 1.0
  - OpenFlow 1.0 only maps well on to (small) TCAM switch tables
    - Tiny fraction of switch functionality
  - Thus, they often artificially constrain topology and/or addressing

Exposed by OpenFlow 1.0 (and thus most of SDN)
SPARTA: Scalable Per-Address RouTing Architecture

- **SPARTA**: Simple, HW-efficient, flexible routing mechanism
  - Build one spanning tree per destination host ([VLAN ID, DMAC])
  - Install one rule per tree per switch in (huge) L2 exact match table

- **Characteristics of SPARTA**
  - Supports arbitrary (connected) physical topology
  - Exploits all available paths (statistically)
  - Leaves TCAMs for designed purposes (security, policy-based routing, …)
  - Flexible framework for traffic engineering, traffic steering, failure recovery, quality of service management, …
Data Center Network Design Goals

- Scalable
  - 10s to 100s of thousands of hosts
- Efficient use of bandwidth
  - Mesh topologies from HPC?
- Efficient host mobility
- Low latency
- Respect layering
- Multi-tenancy
- Very dynamic → self-configuration
- Compatible with existing / planned hardware
- Converged data and storage networks (CEE)
A Brief Tour Through a Modern 10GbE Switch
Modern Switch Hardware Overview

- **Powerful Merchant Silicon Switch Chip**
  - Line rate packet forwarding

- **Embedded Control Plane CPU**
  - Large legacy codebase

- **Lots of 10GbE & 40GbE PHYs**

---

**Gryphon Main Board Block Diagram**

- **Broadcom “Trident” 64 Port 10GbE Switch**
- **Step-Z Mezz Connector #2** (to DC)
- **Step-Z Mezz Connector #1** (to DC)
- **USB**
- **DC QSFP+ LEDs**
- **LED Driver (CPLD2)**
- **RS232**
- **Fan Controller**
- **Fan LEDs**
- **Port LEDs**
- **4 Status LEDs**
TCAMs: Designed for limited use (security ACLs, PBR, …)
L2/FDB table: Huge, plentiful, simple to expand (RAM)
ECMP and multicast tables: Additional flexibility
Basic SPARTA routing

- **Goal**: Route using large L2 table on arbitrary (mesh) topology
- **Solution**: Build spanning tree rooted at each destination
- **All links used → approximate load balancing w/o ECMP**
Constructing SPARTA Routes

- Basic option: Use BFS to build min-length paths
  - Random
  - Weight links by load
  - …
Constructing SPARTA Routes

- Basic option: Use BFS to build min-length paths
  - Random
  - Weight links by load
  - ... 

- Some workloads/topologies benefit from non-min routes

- Non-minimal (NM) PAST
  - Do a BFS from a random switch as the root
  - Change directions on route from root to destination
SPARTA Discussion

- One L2 entry per switch per tree → scales to > 100K hosts
- Consumes no TCAM entries for basic routing
- Obeys layering (does not re-use VLAN tag or other bits)
- Broadcast/multicast: No change → provide via STP or SDN
- Security: Use VLANs as normal (or ACLs)
- Virtualization: Use any higher layer virtualization overlay (e.g., NetLord, SecondNet, MOOSE, VXLAN)
SPARTA Implementation
SPARTA Implementation Details

- **Address detection and resolution:**
  - Uses controller for ARP, DHCP, IPv6 ND, and RS for scalability

- **Route computation:**
  - 8,000 hosts → 40μsecs – 1ms per tree (300ms per network)
  - 100,000 hosts → 500μsecs – 5ms per tree (40s per network)

- **Route installation:**
  - 700-1600 new rules per second per switch
  - 2-12ms rule install latency → eagerly install routes

- **Failure recovery:**
  - Should patch affected portions of trees first
  - Randomly rebuild trees for link joins
SPARTA Performance

- Simulated to allow evaluation at scale
  - Assume max-min TCP fairness to make simulation feasible

- Compared against:
  - STP, Valiant routing, ECMP (multipath routing)

- Workloads:
  - Urand: Uniform random — benign
  - Stride-S: Host i sends to host ((i+S)%N) — adversarial (intra-rack)
  - Shuffle-K: 128MB to all hosts, random order, K active connections
  - MSR: Synthetically generated from MSR data (light load)

- Topologies: Equal bisection bandwidth (oversubscription ratios) of…
  - EGFT (fat tree), Hyper-X (flattened butterfly), Jellyfish (random)
Urand workload on Jellyfish

PAST performs as well as ECMP multipath routing

Spanning Tree performs terribly
Stride workload on Jellyfish

Non-Minimal PAST performs better than Valiant load balancing

Spanning Tree performs terribly
Summary for SPARTA

- Meets all of our requirements for a DCN by exploiting only the most basic Ethernet forwarding hardware

- Scalable, low-latency, high-bandwidth network from COTS ToR switches (So we can exploit HPC-style mesh topologies!)

- Can provide 1-2X performance of ECMP

- Implemented on existing hardware w/ OF 1.0 (!!!)

- Leaves TCAM entries for designed uses: PBR, security, …

- Flexible framework for traffic engineering, traffic steering, QoS management, resiliency, …

- For full results, see CoNEXT 2012 paper (next week)
Suggestions for SDN Research

- Understand and exploit what is in the actual hardware
  - Do not let OpenFlow specification restrict your vision...
  - ... but don’t assume magical hardware ("unicorns and rainbows")

- Consider what can be done by running “SDN-aware” functions on the control processor (ala HP Labs’ DevoFlow)
  - Controller understands “big picture” \(\rightarrow\) guides switch-local decisions
  - Switch firmware can respond in \(\mu\)secs, not msecs
  - Opportunity: Indigo or similar open source OpenFlow switch firmware
  - Pushing it to the limit \(\rightarrow\) switchlets (Active Networking reborn?)

- Why just networks? Software-defined everything
  - SDS: software-defined storage (lots of startups claiming this)
  - SDC: software-defined computation (VMs kind of do this)
  - SDDC: software-defined data center
TCP Bolt: Faster small flows with lossless Ethernet

- Lossless ➔ no congestion collapse
- Send at line-rate immediately
- 1.5-3X better than vanilla TCP for 64K–8M
  – many real DC flows are this size