NaaS

Network-as-a-Service in the Cloud

Paolo Costa
costa@imperial.ac.uk

joint work with
Matteo Migliavacca, Peter Pietzuch, and Alexander L. Wolf
Motivation

Mismatch between app. abstractions & network

• How the programmers see the network
Motivation

*Mismatch between app. abstractions & network*

• How the network really looks like
Motivation

Mismatch between app. abstractions & network

- What programmers really see of the network

```c
int send(int sockfd, const void *msg, int len, int flags);
```

```c
int recv(int sockfd, void *buf, int len, int flags);
```

- No control over network resources
- Hard to map distributed apps onto the physical topology
Example #1: MapReduce

- Many cloud data centers have high degree of oversubscription (e.g., 1:20 [IMC’10])
  - Intra-rack bandwidth >> inter-rack bandwidth

- Location of map and reduce tasks is critical

70% of cross track traffic is reduce traffic

50% reduce phases takes 62% longer than ideal placement

Source: Ananthanarayanan et al. OSDI’10
Example #1: MapReduce

- Many cloud data centers have high degree of oversubscription (e.g., 1:20[IMC’10])
  - Intra-rack bandwidth >> inter-rack bandwidth

- Location of map and reduce tasks is critical

- Current approach
  - *Reverse-engineer the network*
    Combination of low-level tools (ping, traceroute, ...) and complex clustering algorithms

- Issues
  - Low-level process
  - Time consuming
  - Potentially inaccurate
Example #1: MapReduce

- Many cloud data centers have high degree of oversubscription (e.g., 1:20[IMC’10])
  - Intra-rack bandwidth >> inter-rack bandwidth

Wish list #1: Network Visibility
Tenants are provided with an (abstract) view of their allocated VMs
No need for reverse-engineering, easier deployment

Issues
- Low-level process
- Time consuming
- Potentially inaccurate
Example #2: Iterative Jobs

- Iterative jobs often adopt an one-to-many communication pattern
  - e.g., Netflix Collaborative Filtering

- Current approach
  - Point-to-point
  - Application-level multicast tree
  - BitTorrent-like solutions

Source: Chowdhury et al. SIGCOMM’11
Example #2: Iterative Jobs

- Iterative jobs often adopt an one-to-many communication pattern
  - e.g., Netflix Collaborative Filtering

- Current approach
  - Point-to-point
  - Application-level multicast tree
  - BitTorrent-like solutions

- Issues
  - The server 1Gbps link is the bottleneck
  - Even perfect network visibility would not help
Example #2: Iterative Jobs

- Iterative jobs often adopt an approach that creates a large number of small iterations.

**Wish list #2: Custom Forwarding**

Tenants can implement custom routing protocols
E.g., anycast, multicast, content-based routing, key-based routing, multi-path routing, ...

**Issues**
- The server 1Gbps link is the bottleneck
- Even perfect network visibility would not help
Example #3: Interactive Queries

- Many large-scale web services use a **partition-aggregate** pattern
  - Queries are sent to multiple workers and responses are combined

- **Current approach**
  - Responses from workers are aggregated at intermediate layers
  - E.g., Google Search

Source: Jeff Dean. WSDM’09
Example #3: Interactive Queries

- Many large-scale web services use a partition-aggregate pattern
  - Queries are sent to multiple workers and responses are combined

- Current approach
  - Responses from workers are aggregated at intermediate layers
  - E.g., Google Search

- Issues
  - Requires high network bandwidth
  - Aggregator servers become the bottlenecks
  - Custom forwarding would not help
Example #3: Interactive Queries

- Many large-scale web services use a partition-aggregate pattern

Wish list #3: In-network Processing

Tenants can perform arbitrary packet processing on path E.g., in-network aggregation [Camdooop@NSDI’12], opportunistic caching, semantic de-duplication

- Requires high network bandwidth
- Aggregator servers become the bottlenecks
- Custom forwarding would not help
Introducing NaaS

• **Goal**
  – *Mechanisms and abstractions to enable cloud tenants to efficiently, easily, and safely process packets within the network*

• This entails *visibility* over network resources, *custom forwarding* and *processing* of packets

• Providers would benefit too
  – Today they also need to reverse-engineer applications
  – NaaS would allow more fine-grained traffic engineering

• This is complementary to...
  – SDN / OpenFlow / ...
  – Focus on *application-specific* rather than *application-agnostic* services

• ...but some techniques can be re-used
Why Now?

- **DCs are not mini-Internets**
  - Single owner / administration domain
  - We know (and define) the topology
  - Low hardware and software (network protocols) diversity
  - Trusted components (e.g., hypervisors)

- Several proposals for **software-based routers**
  - RouteBricks, ServerSwitch, PacketShader, SideCar, NetMap, ...

- Typically, these are used to replace traditional (**application-agnostic**) network services (e.g., IPv4 forwarding, DPI)

- **Why don’t use them also to implement** **application-specific services?**
  - E.g., aggregate packets in a distributed query or content-based routing
NaaS Architecture

• Switches are augmented with processing capabilities
  – Software routers a la Routebricks or hybrid solutions like ServerSwitch

• (Oversubscribed) Fat-tree-like topology
  – Lower in-bound switch throughput
  – E.g., for a 27K-server, max throughput is 48 Gbps
NaaS Architecture

- Switches are augmented with processing capabilities (**NaaS box**)
  - Software routers a la Routebricks or hybrid solutions like ServerSwitch

- (Oversubscribed) Fat-tree-like topology
  - Lower in-bound switch throughput
  - E.g., for a 27K-server, max throughput is 48 Gbps

- Tenants deploy their processing elements (**INPE**) on each NaaS box
  - Fast-path for non-NaaS traffic
(Preliminary) Evaluation

• **Questions:**
  - What are the benefits for NaaS users?
  - What is the impact for non-NaaS users?
  - What is the processing rate required?

• **Setup**
  - Flow-level simulator
  - 8,192-server fat-tree topology (32-Gbps switches)
  - 80% traditional TCP flows, 20% combination of multicast, aggregation, caching
Total flow completion time

R=1 Gbps already reduces time by 65%

96% time reduction

Faster NaaS boxes

NaaS box maximum processing rate in Gbps (R)

Time normalized to baseline

Worse

Better

Paolo Costa
Total flow completion time

Even a low processing rate is enough to achieve significant benefits

R=1 Gbps already reduces time by 65%

96% time reduction

Even a low processing rate is enough to achieve significant benefits
Individual Flow Completion Time

This includes non-NaaS flows too
The use of NaaS is beneficial for all flows (including non-NaaS ones)

This includes non-NaaS flows too
Challenges

- **Scalability and performance isolation**
  - Traditional software routers assume **handful of trusted services**
  - In NaaS we expect 10s or 100s of (potentially malicious or poorly written) INPEs per switch

- **Programming abstractions**
  - We should not expose the actual network programming
    - Too complex for many users
    - Sensitive information
  - Trade-off between flexibility and performance

- **Pricing schemes**
  - How we should charge tenants?
Summary

• Currently tenants have little control over the network

• NaaS focuses on enabling tenants to deploy applications within the network
  ✓ Efficiency
  ✓ Simplified development
  ✓ Providers benefit too

• On-going work
  – SideCar[HotNets‘11] inspired design
  – Transparent acceleration of mainstream applications