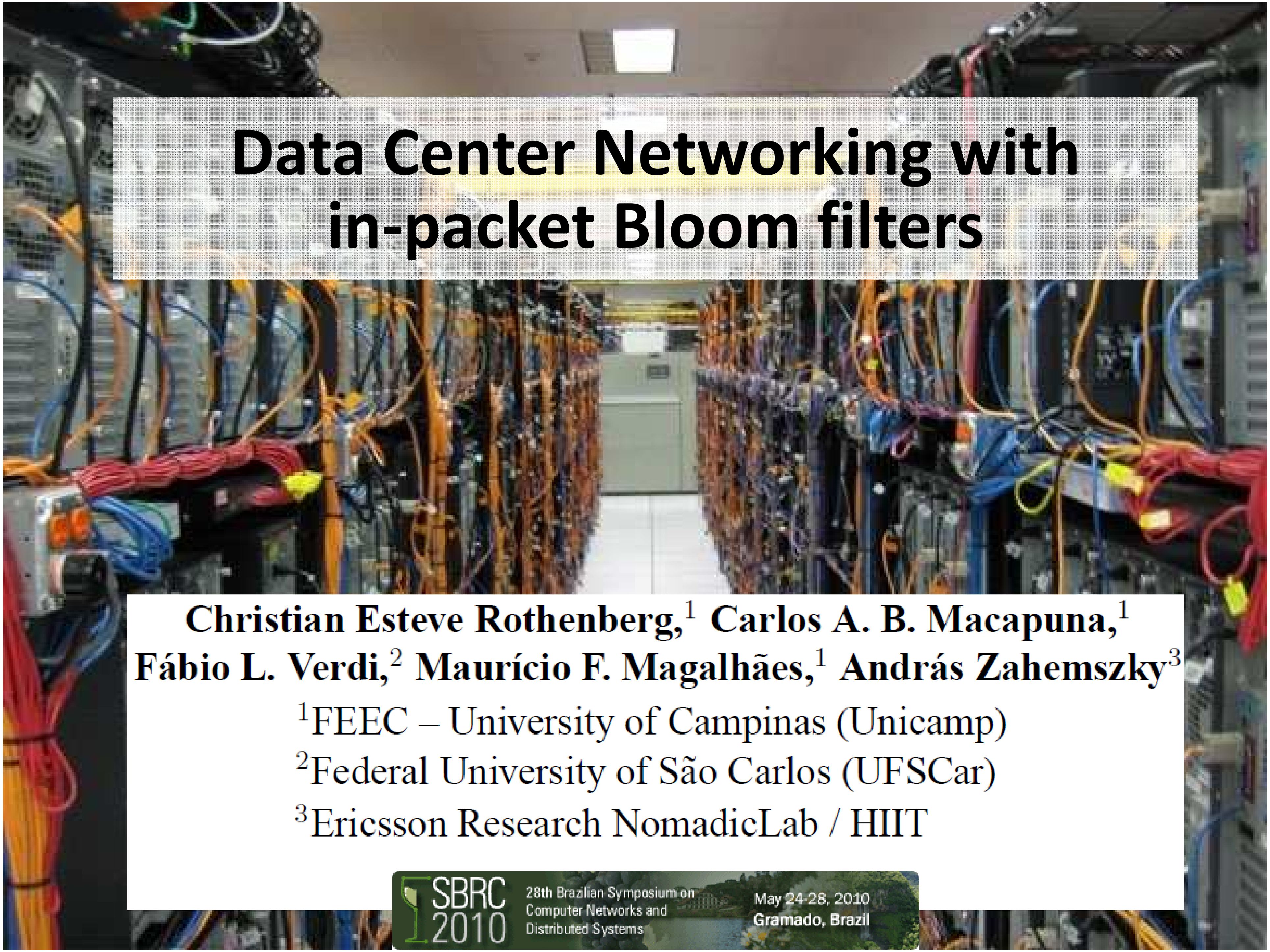


Data Center Networking with in-packet Bloom filters

Christian Esteve Rothenberg,[1] Carlos A. B. Macapuna,[1] Fábio L. Verdi,[2] Maurício F. Magalhães,[1] András Zahemszky[3]

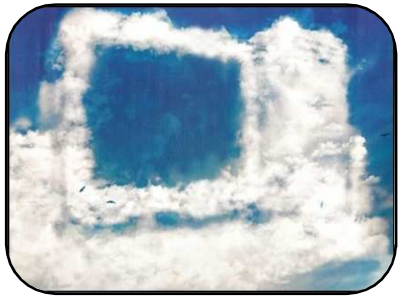
[1]FEEC – University of Campinas (Unicamp)

[2]Federal University of São Carlos (UFSCar)

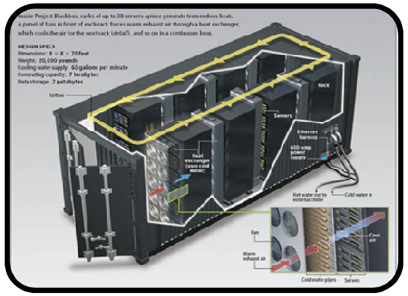
[3]Ericsson Research NomadicLab / HIIT

SBRC 2010
28th Brazilian Symposium on Computer Networks and Distributed Systems
May 24-28, 2010
Gramado, Brazil

Agenda



Motivation

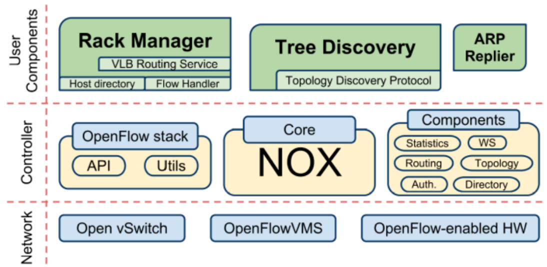


New data center designs

Requirements

SiBF: Switching with in-packet Bloom filters

Design principles



Implementation

Table 1. Evaluation of the state requirements in terms of entries at switches.

Physical hosts	2,880			23,040			103,608		
Racks	144			1152			5184		
Aggr. Switches	24 ($p_1 = 24$)			96 ($p_1 = 48$)			144 ($p_1 = 144$)		
Core Switches	12 ($p_2 = 24$)			24 ($p_2 = 96$)			72 ($p_2 = 144$)		
	VL2	Portland	SiBF	VL2	Portland	SiBF	VL2	Portland	SiBF
Entries at ToR	200	120	120	1292	120	120	5420	120	120
Entries at AGGR	180	24	24	1272	48	48	5400	144	144
Entries at CORE	180	24	24	1272	96	96	5400	144	144

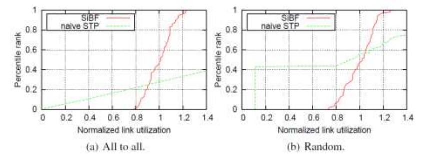


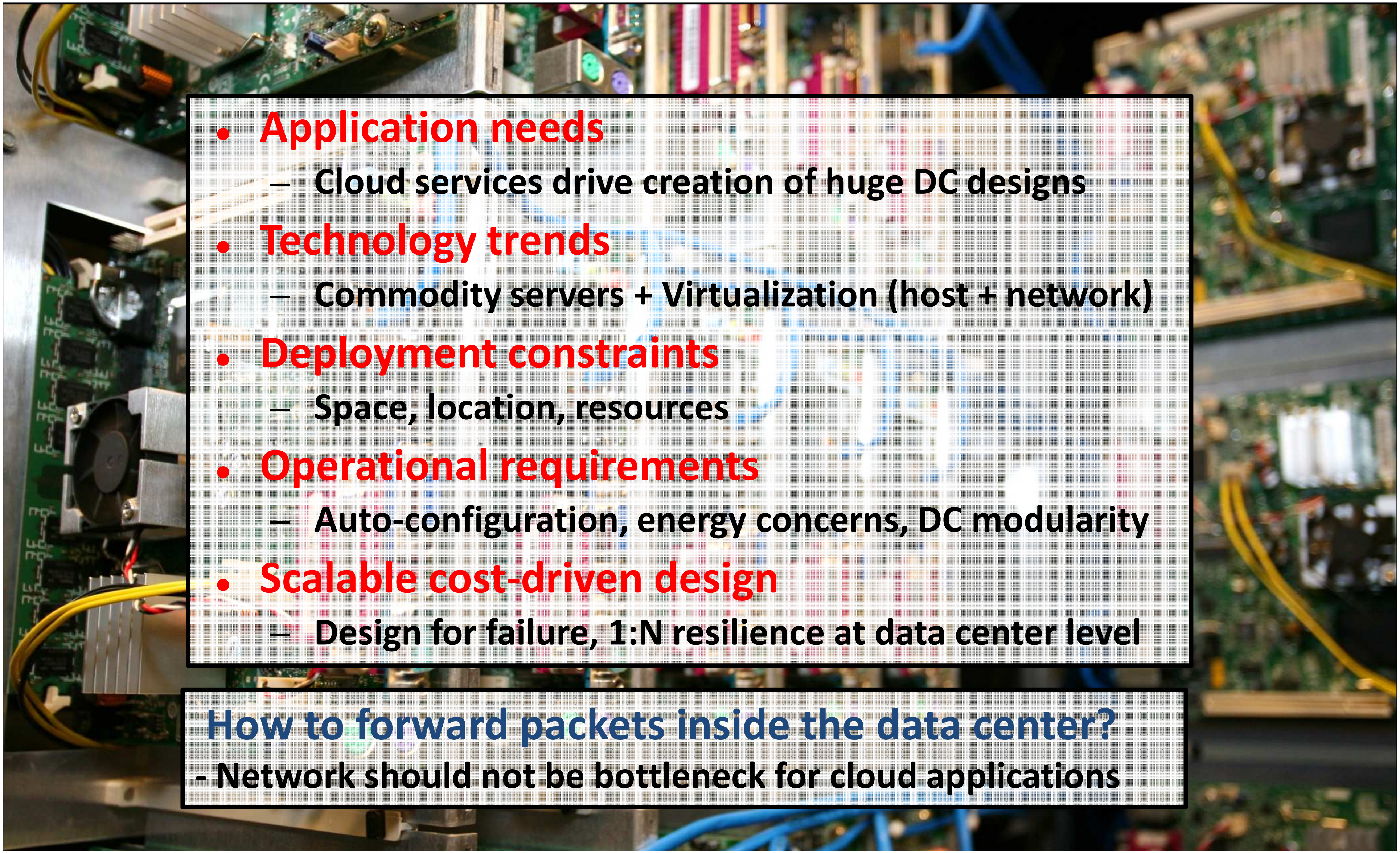
Figure 5. Evaluation of the load balancing behaviour. CDFs of the link utilization.

Evaluation

Future work

Conclusions

New data center design drivers

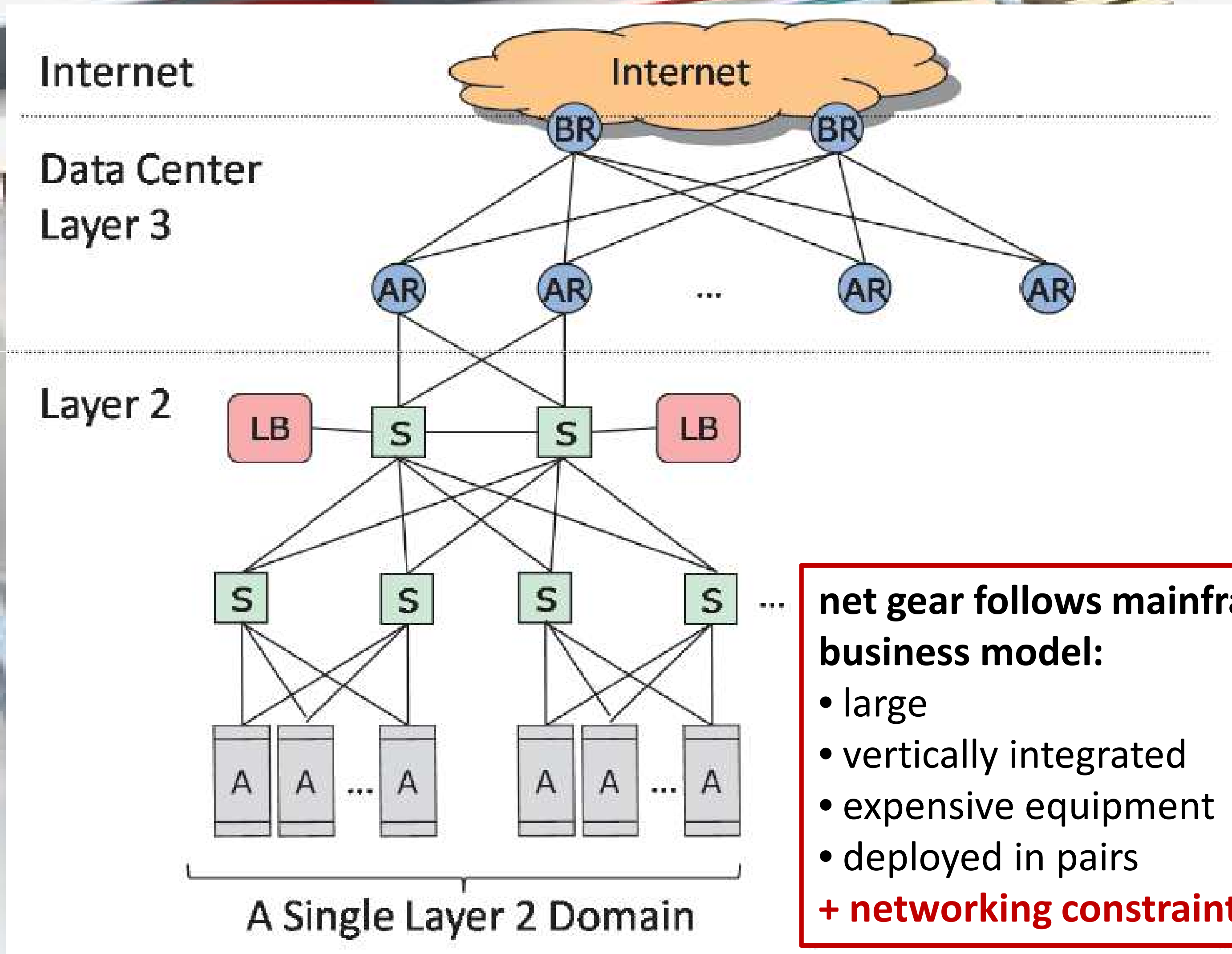


- **Application needs**
  - Cloud services drive creation of huge DC designs
- **Technology trends**
  - Commodity servers + Virtualization (host + network)
- **Deployment constraints**
  - Space, location, resources
- **Operational requirements**
  - Auto-configuration, energy concerns, DC modularity
- **Scalable cost-driven design**
  - Design for failure, 1:N resilience at data center level

How to forward packets inside the data center?

- Network should not be bottleneck for cloud applications

Traditional DCN architectures (Cisco view)



net gear follows mainframes business model:

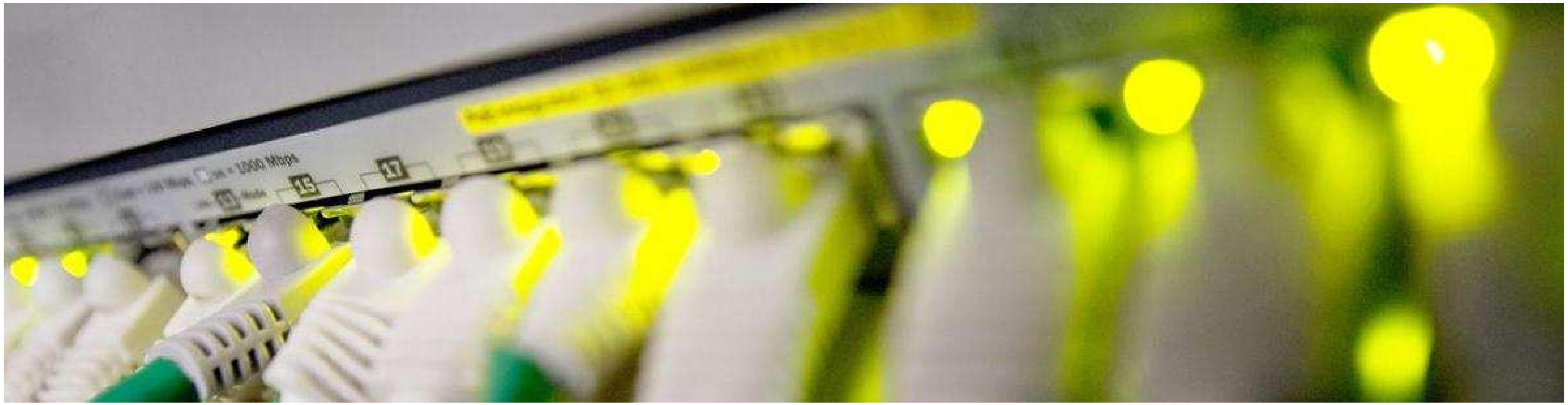
- large
- vertically integrated
- expensive equipment
- deployed in pairs

+ networking constraints !

Some issues with conventional DC designs

Networking constraints of traditional L2/L3 hierarchical organization:

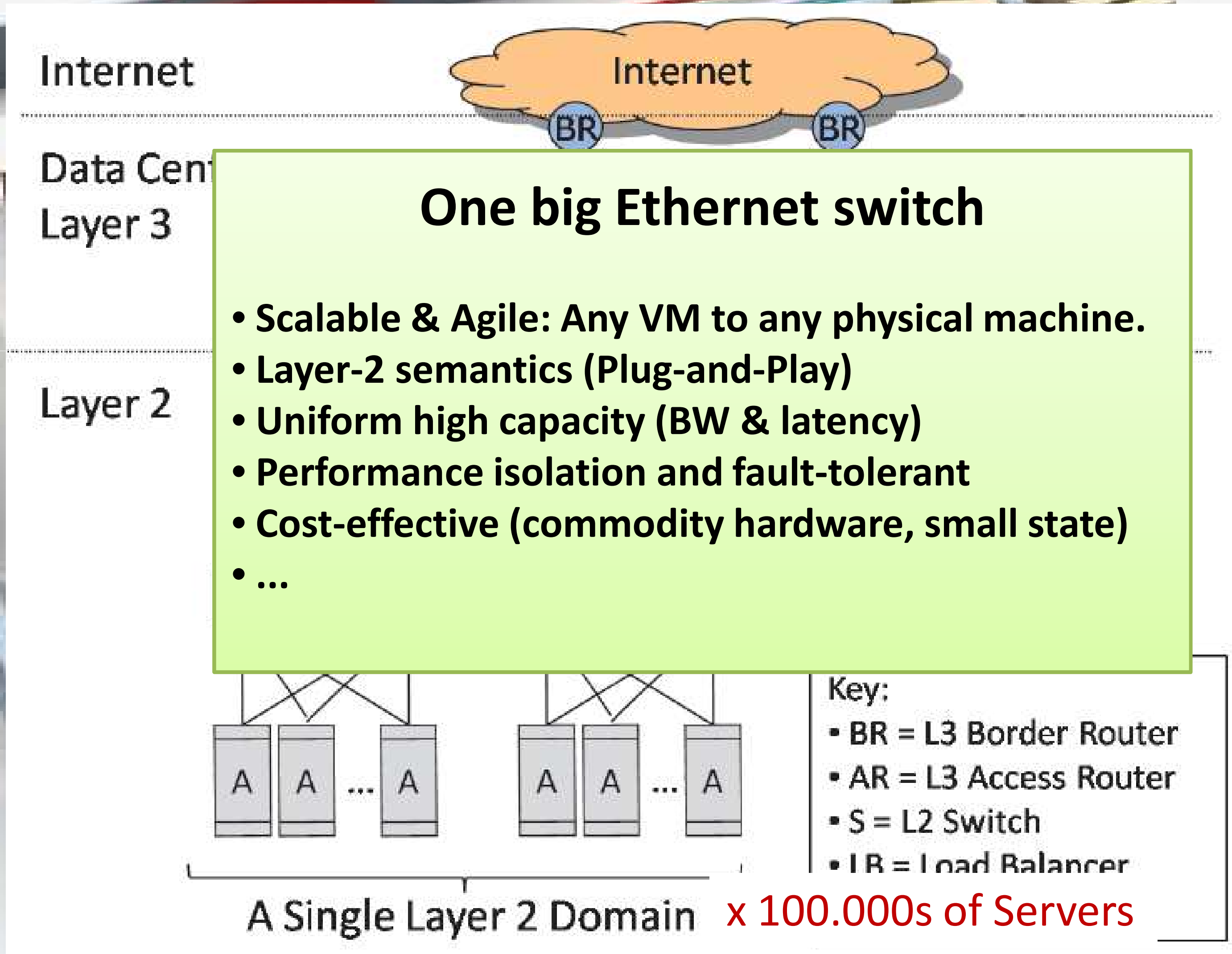
- Fragmentation of resources (VLAN, subnetting)
- Limited server-to-server capacity (high oversubscription)
- Ethernet scalability (FIB size, STP, flooding, ARP broadcast)
- Low performance under cloud application traffic patterns
- Reliability: 2 is a poor choice for redundancy at scale



Ideal DCN from a Cloud App dev view

One big Ethernet switch

- Scalable & Agile: Any VM to any physical machine.
- Layer-2 semantics (Plug-and-Play)
- Uniform high capacity (BW & latency)
- Performance isolation and fault-tolerant
- Cost-effective (commodity hardware, small state)
- ...



Related work

VL2 [SIGCOMM´09]

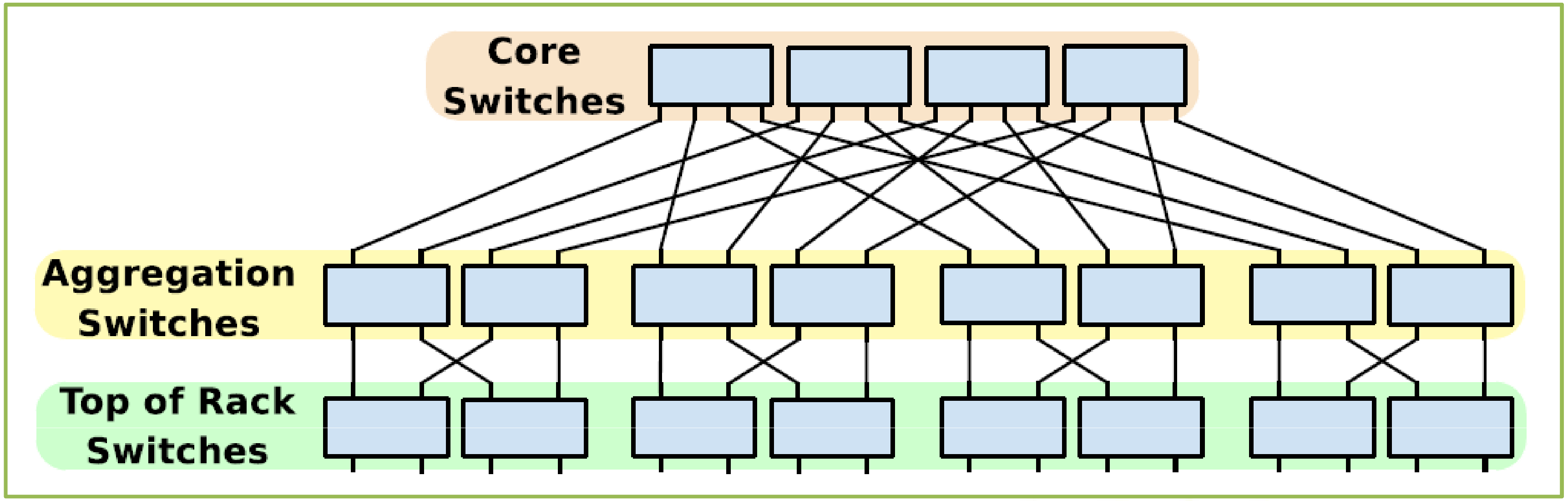
- Layer 3 routing fabric used to implement a virtual layer 2
- Unmodified switch hardware and software
- End hosts modified to perform enhanced resolution to assist routing and forwarding (IP-in-IP source routing)

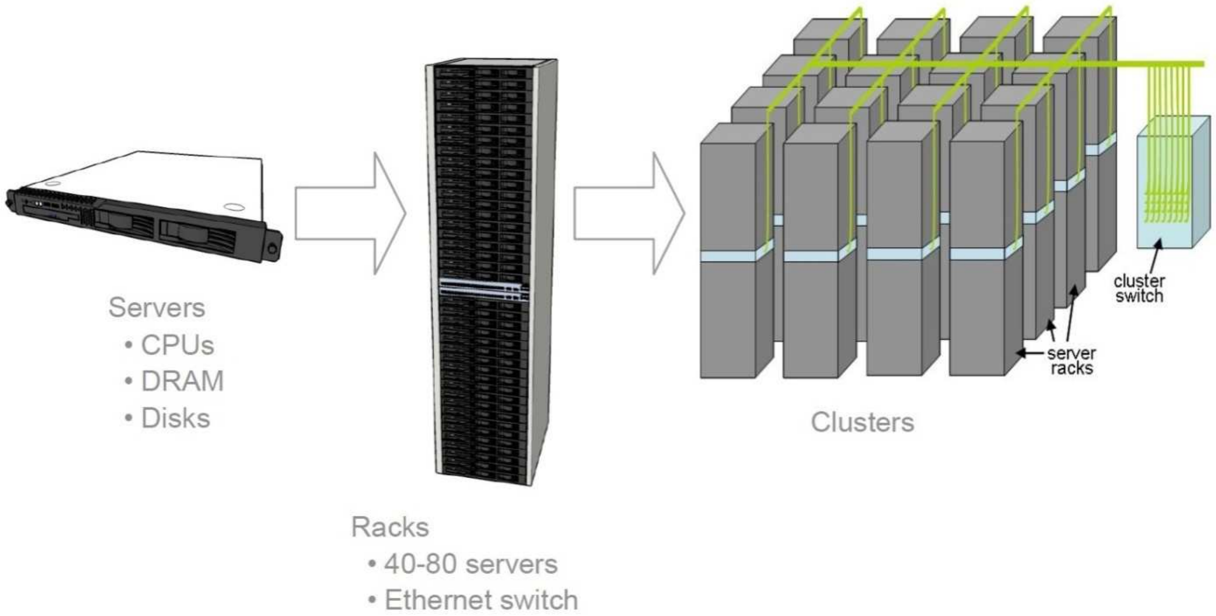
Portland [SIGCOMM´09]

- Separates host identity from host location
  - Uses IP address as host identifier
  - Introduces "Pseudo MAC" (PMAC) addresses internally to encode endpoint location
- Runs on commodity switch hardware with OpenFlow API

BCUBE and more to come…

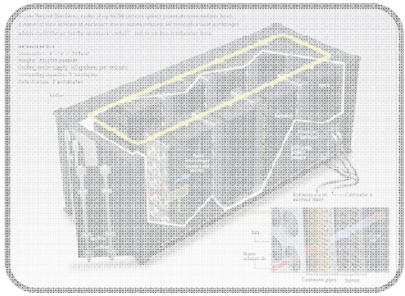
New generation DCN topologies





Agenda

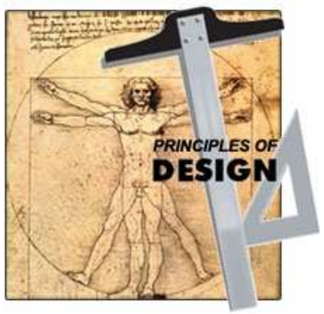
Motivation



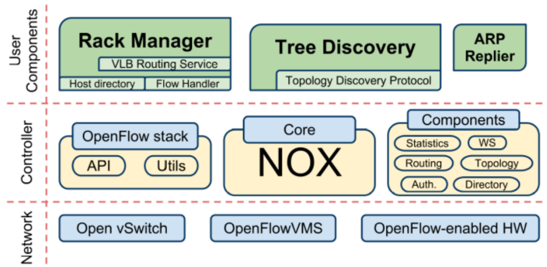
New data center designs

Requirements

SiBF: Switching with in-packet Bloom filters



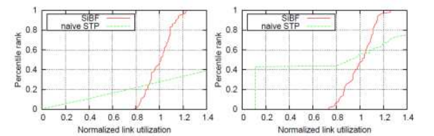
Design principles



Implementation

Table 1. Evaluation of the state requirements in terms of entries at switches.

Physical hosts	2.880			23.040			103.608		
Racks	144			1152			5184		
Aggr. Switches	24 ($p_1 = 24$)			96 ($p_1 = 48$)			144 ($p_1 = 144$)		
Core Switches	12 ($p_2 = 24$)			24 ($p_2 = 96$)			72 ($p_2 = 144$)		
	VL2	Portland	SiBF	VL2	Portland	SiBF	VL2	Portland	SiBF
Entries at ToR	200	120	120	1292	120	120	5420	120	120
Entries at AGGR	180	24	24	1272	48	48	5400	144	144
Entries at CORE	180	24	24	1272	96	96	5400	144	144



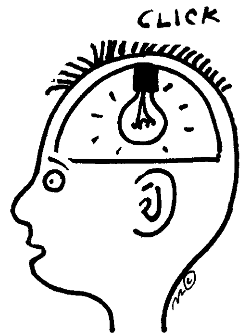
(a) All to all. (b) Random.

Figure 5. Evaluation of the load balancing behaviour. CDFs of the link utilization.

Evaluation

Future work

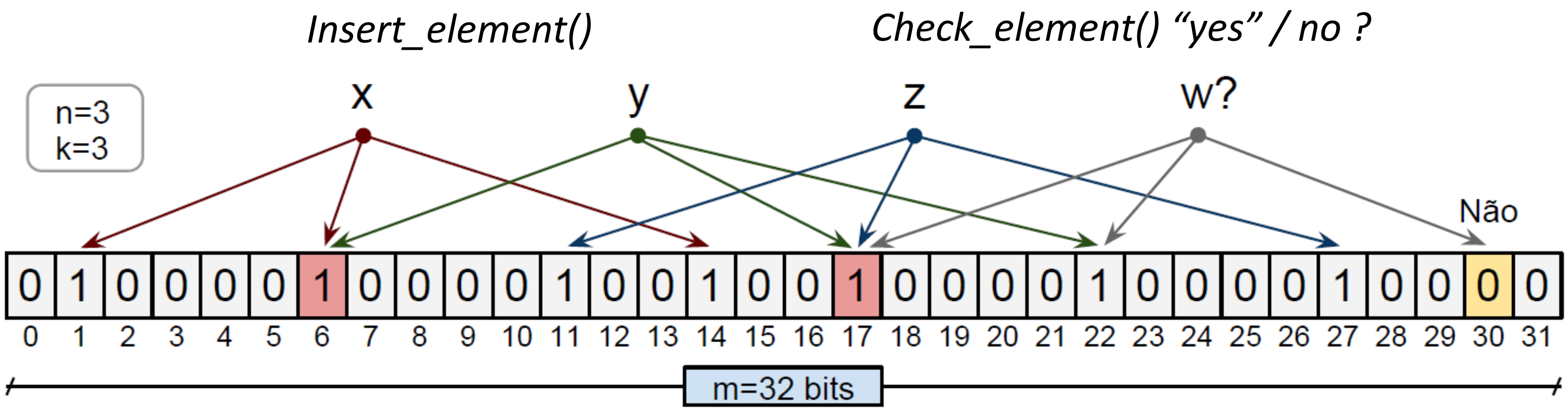
Conclusions



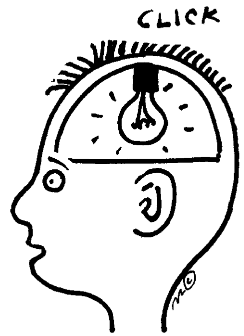
Basic idea

- Compactly represent a *source route* into an in-packet Bloom filter (iBF)
- Carry the 96-bit iBF in the *source* and *destination MAC fields* (MAC re-writing at source and destination ToR switches)
- *Stateless* forwarding by querying next-hop switches in the iBF

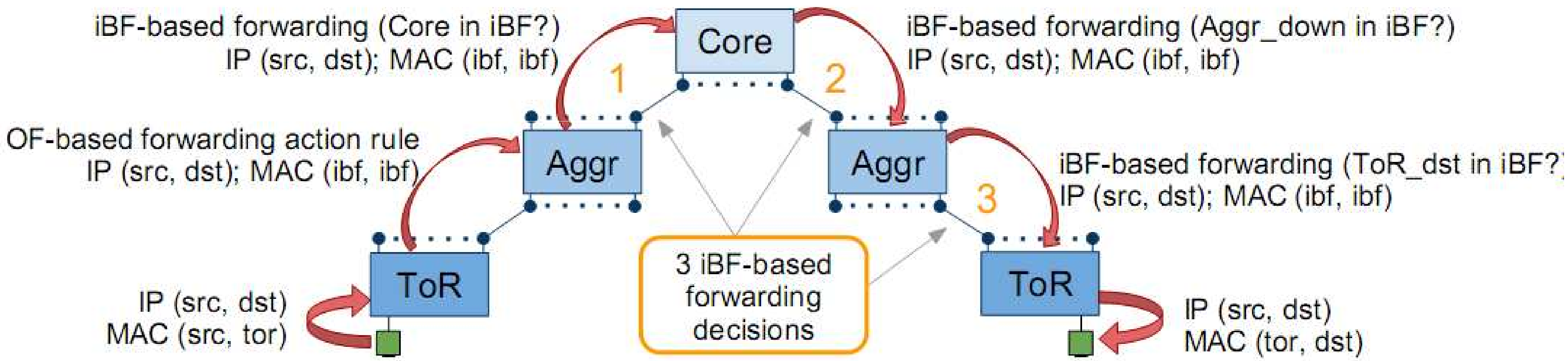
- Bloom filter fundamentals
  - ***m*** bit array — 96 bits of Ethernet SA and DA
  - ***k*** independent hash functions — 7
  - ***n*** elements inserted — 3 MAC addresses (CORE, AGGR and ToR)



Basic idea



In-packet Bloom filter (iBF) based forwarding*:



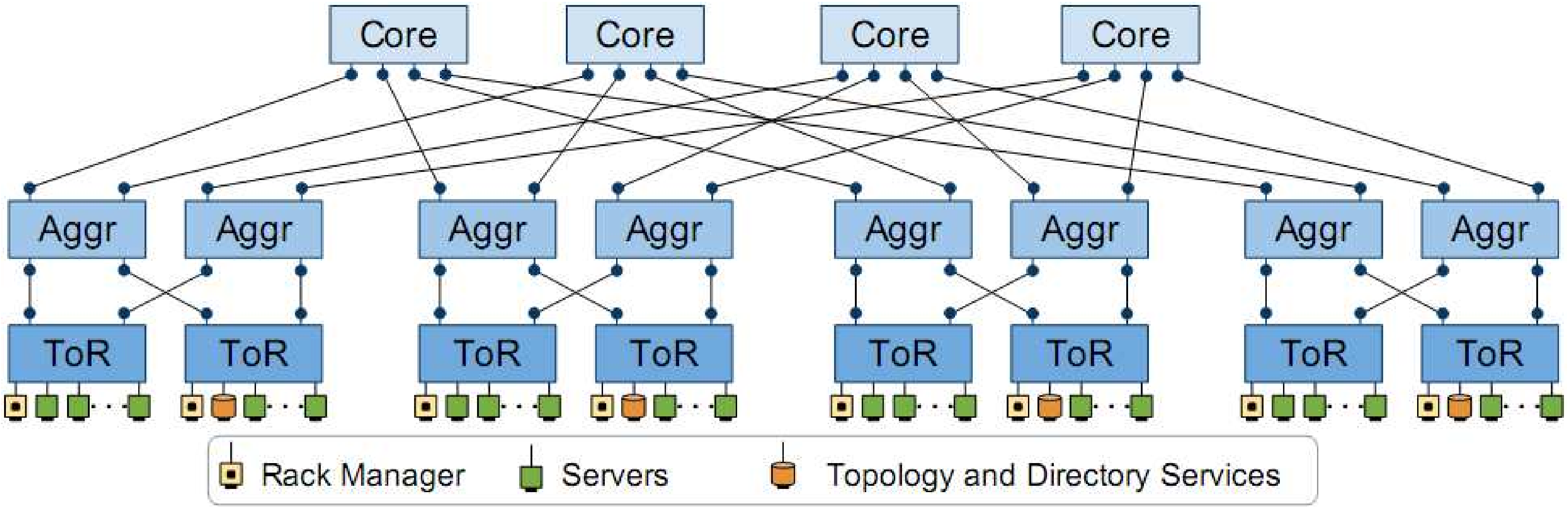
* Jokela, P., Zahemszky, A., Esteve Rothenberg, C., Arianfar, S., and Nikander, P. (2009). LIPSIN: line speed publish/subscribe inter-networking. In *SIGCOMM '09*. ACM.

Design Principles

- **Separating Names from Locations**
  - IP for VM identification, pure “L2” connectivity
- **Source explicit routing**
  - Stateless intermediate switching based on the iBF
- **Direct network control and logically centralized directory**
  - Rack Managers install flows at ToRs and maintain topology and VM dir.
- **Load balancing through path randomization**
  - Exploit path multiplicity to provide *oblivious routing* (i.e., traffic independent randomized packet routing) [VLB]
- **Unmodified end-points and plug & play**
  - Legacy servers and applications are supported off-the-shelf.
  - Auto-configuration of end-hosts and switches (Role Discovery Protocol)
- **Design to cope with failures**
  - Assume any component will fail (built-in fault-tolerance)

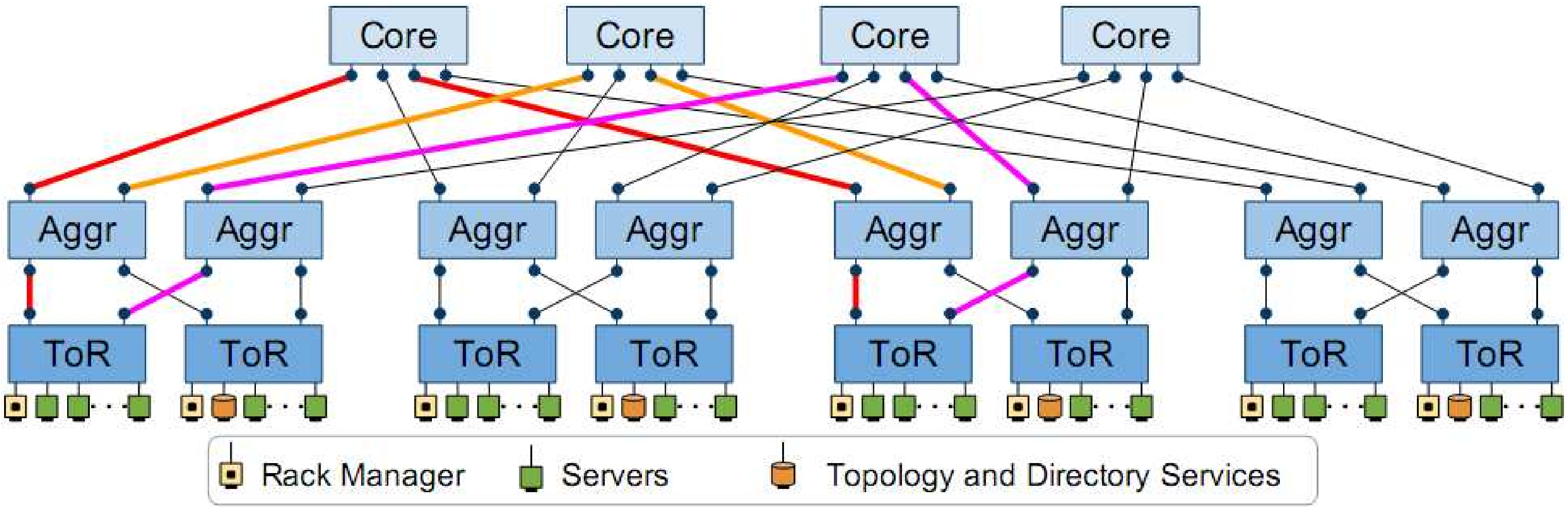
SiBF architecture

- An army of Rack Managers with distributed Topology and Directory services



Valiant Load Balancing

- Random path selection (per-flow)
  - Choose Aggr1, Core, Aggr2
  - iBF encodes Core, Aggr2, ToR



Role Discovery Protocol

Goal: Discovery and auto-configuration of switches

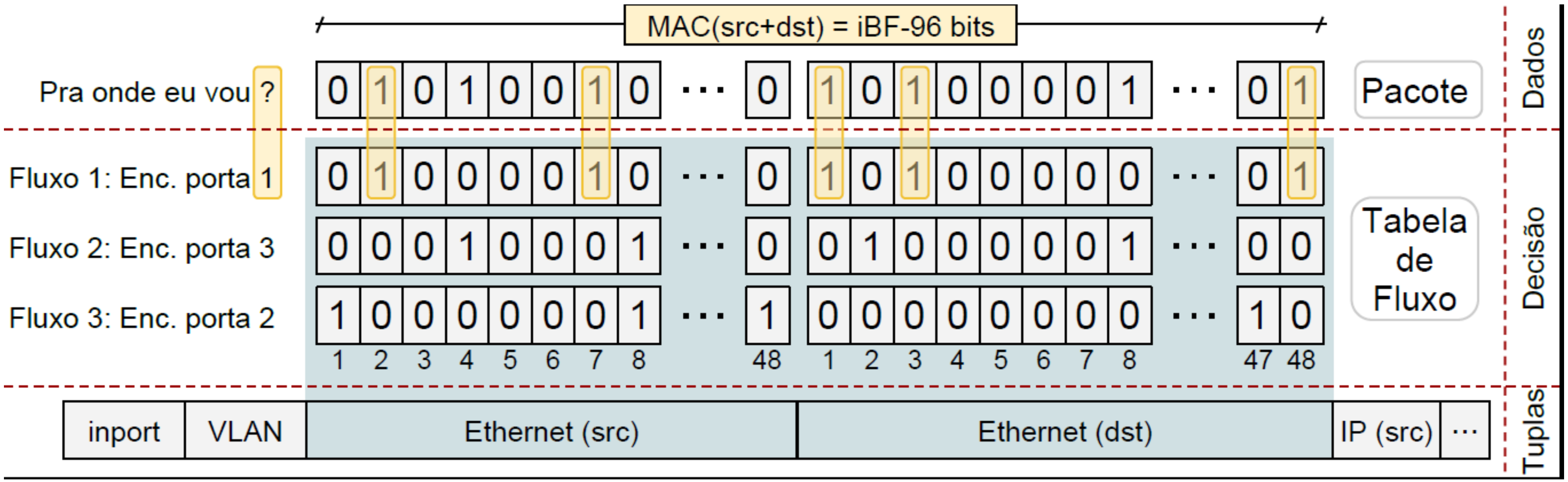
Algorithm 1: Role Discovery Protocol.

```
begin switch_join
    ROLE ← UNDEFINED;
    SendAllPorts(llpd, ROLE);
end
begin arp_receive_server
    if ROLE != TOR then
        ROLE ← TOR;
    end
end
begin lldp_receive_neighbors
    NBROLE ← neighbors.ROLE;
    if NBROLE = (CORE or TOR) then
        ROLE ← AGGR;
    else if NBROLE = AGGR then
        ROLE ← CORE;
    end
end
```

- Similar to the discovery protocol of Portland but simpler
- Leverages the 3-tier topology
- Implemented with TLV extension to LLDP
- Upon neighbor discovery
  - Switch installs neighboring Bloomed MACs entries: *k “hashes”* of the MAC

OpenFlow-based iBF implementation

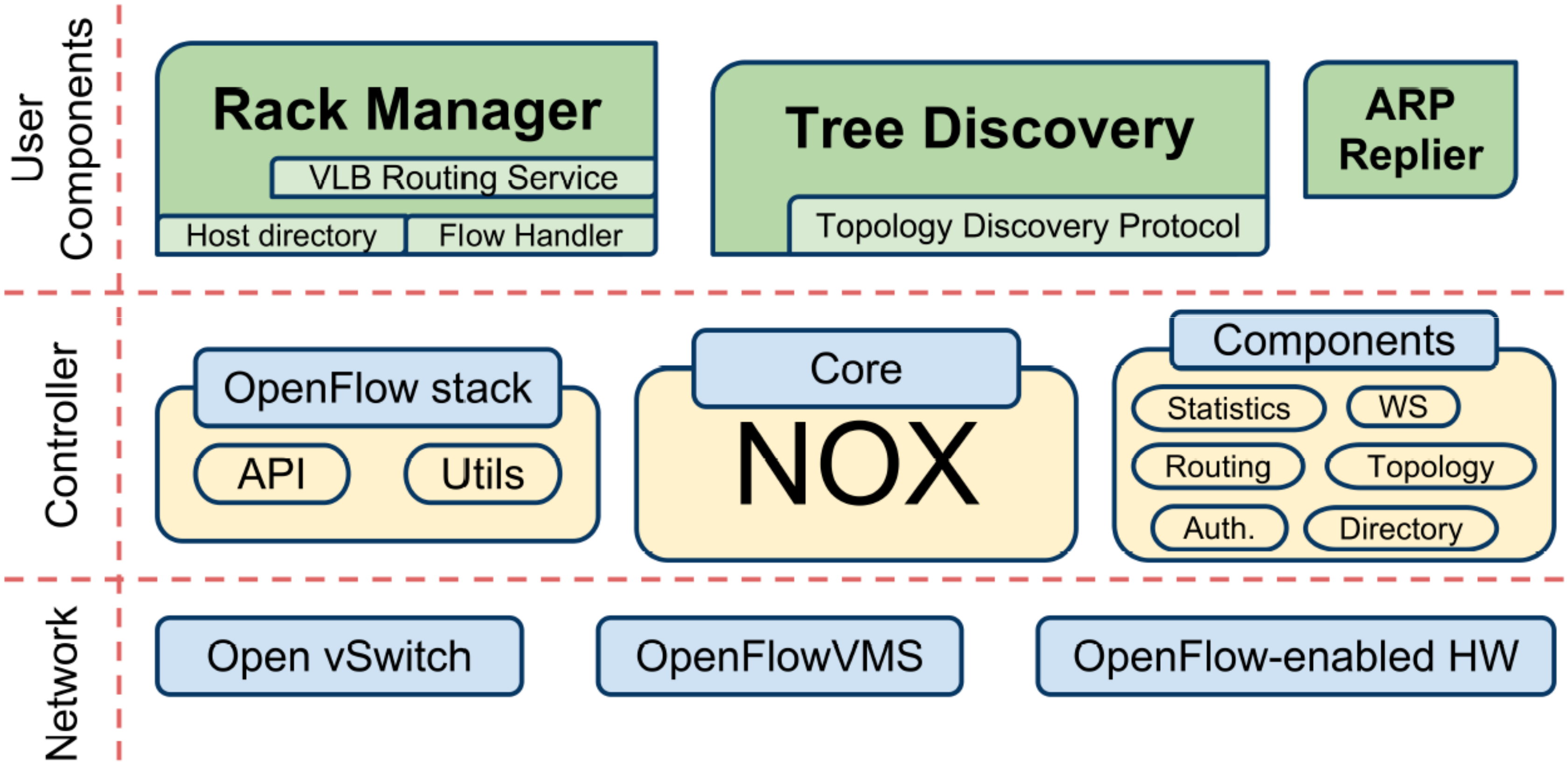
- OpenFlow extension to match on arbitrary wildcarded bit masks
  - Easy to implement: 2 lines of code in the flow matching function
  - Official support expected in upcoming OpenFlow versions



False-positive-free forwarding on Bloomed MAC identifiers

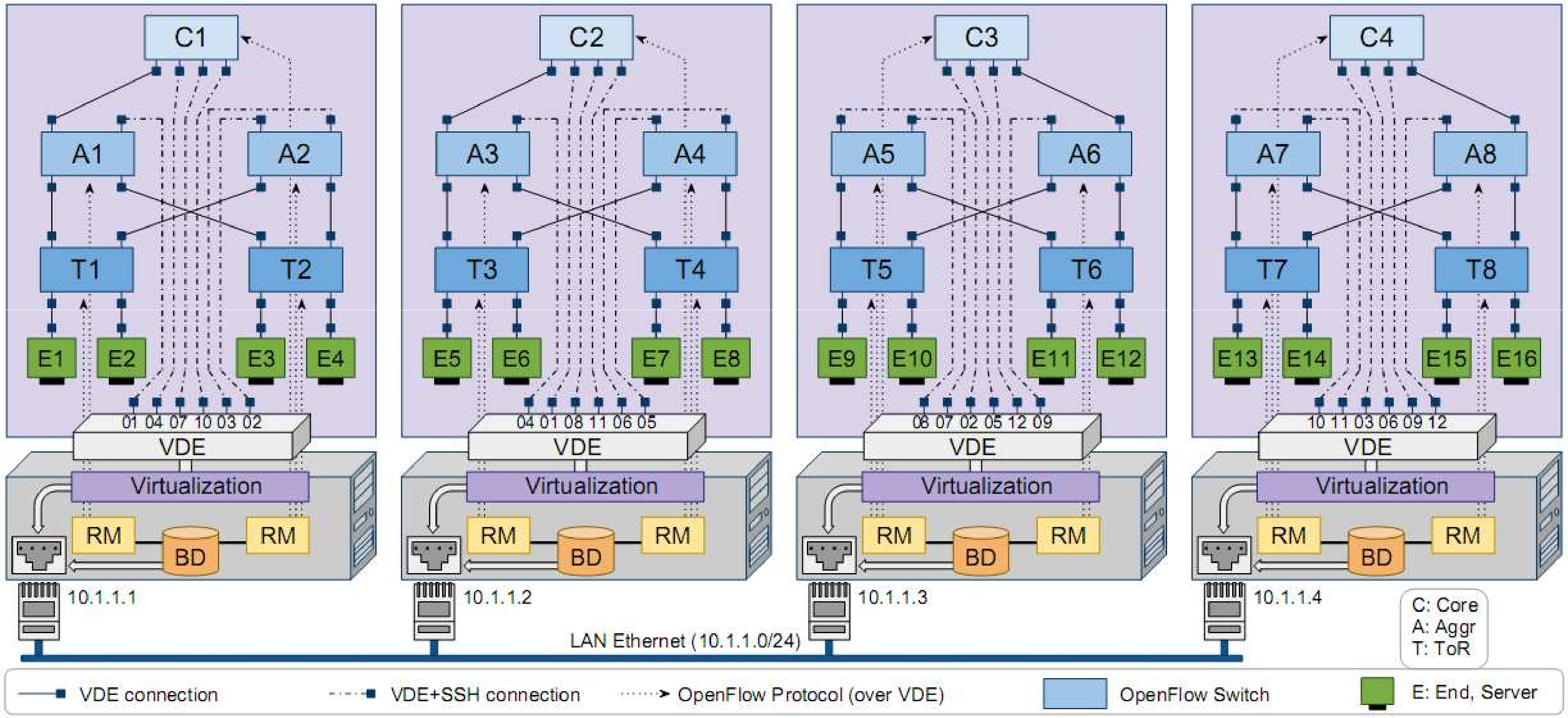
- Instead of traditional exact matching on MAC_{dst}, each forwarding entry contains a 96-bit mask with only *k* 1s based on "hashes" of the neighbouring switch MAC.
- Well-known caveat of Bloom filters: *false positives*
  - 2 or more switches appear as next hop candidates:
    (i) multi-cast the packet along matching interfaces
    (ii) pick one and "pray" (+ temporal fix by controller)
- (iii) Test iBFs for false positives prior to their use!
  - *power of choices* along two dimensions:
    (1) multiple paths, and (2) multiple iBF representations
- RM maintains a ToR_{src}-ToR_{dst} matrix filled only with *false-positive-free iBFs* for the multiple paths

RM controller implementation



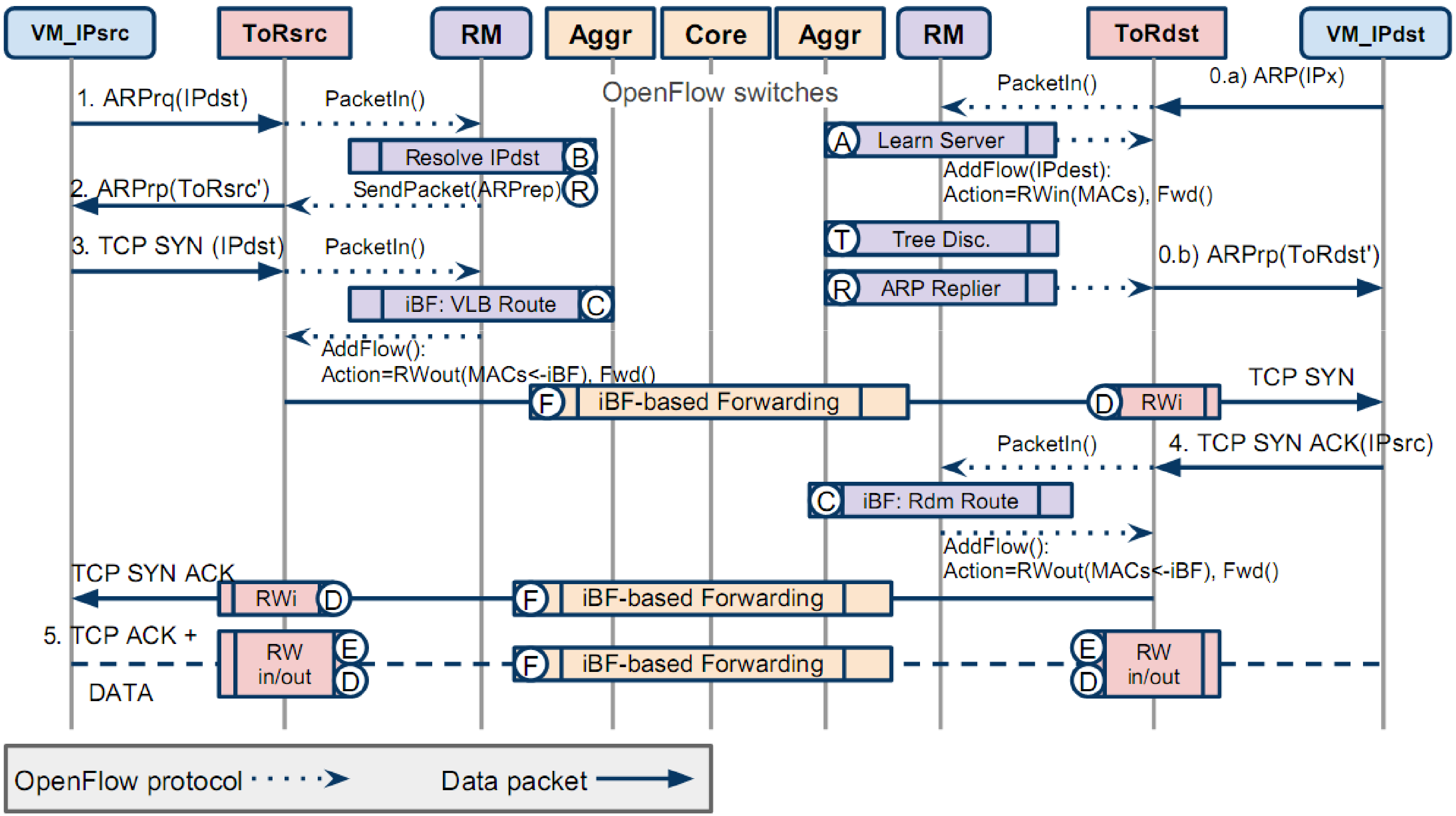
See details of the Distributed Rack Manager implementation in WGCA' 10

Testbed

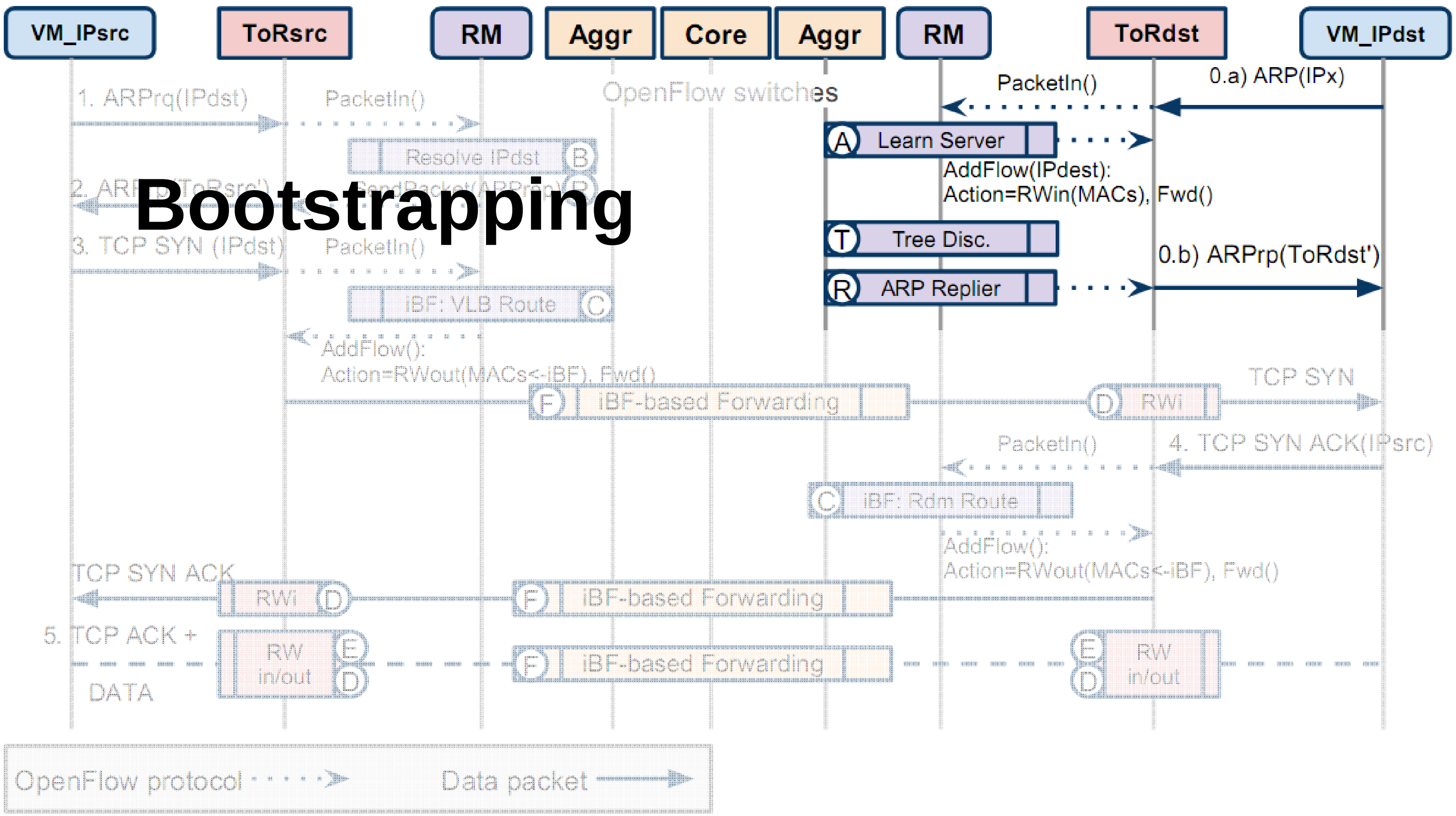


C1
C2
C3
C4
A1
A2
A3
A4
A5
A6
A7
A8
T1
T2
T3
T4
T5
T6
T7
T8
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10
E11
E12
E13
E14
E15
E16
01 04 07 10 03 02
04 01 08 11 06 05
08 07 02 05 12 09
10 11 03 06 09 12
VDE
Virtualization
RM
BD
RM
10.1.1.1
10.1.1.2
10.1.1.3
10.1.1.4
LAN Ethernet (10.1.1.0/24)
C: Core
A: Aggr
T: ToR
VDE connection
VDE+SSH connection
OpenFlow Protocol (over VDE)
OpenFlow Switch
E: End, Server

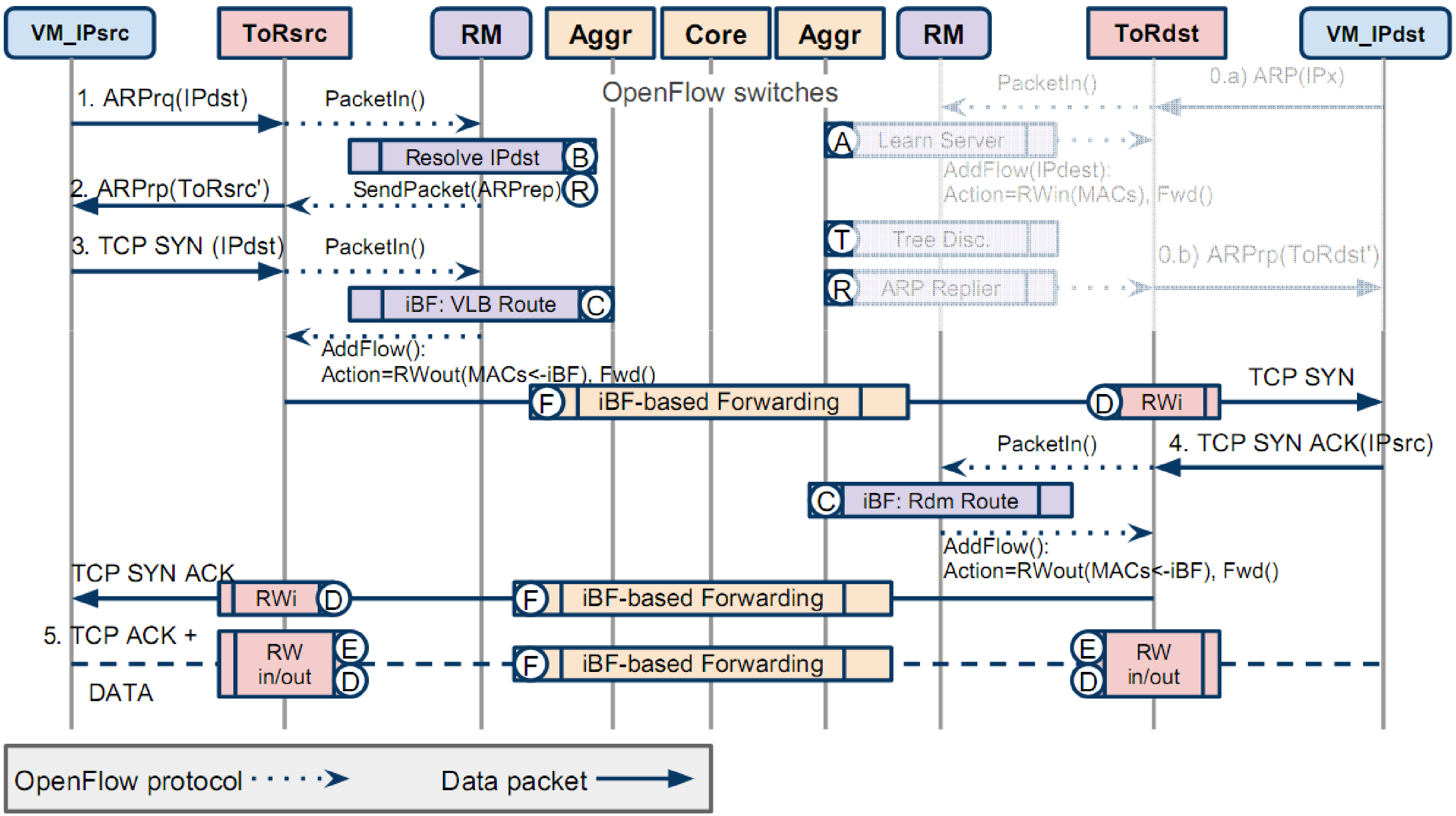
Message diagram



Message diagram

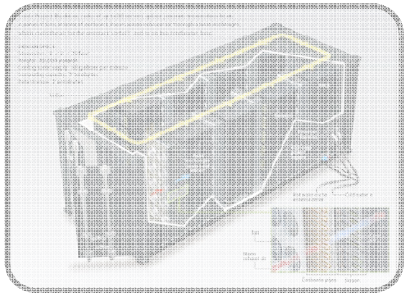


Message diagram



Agenda

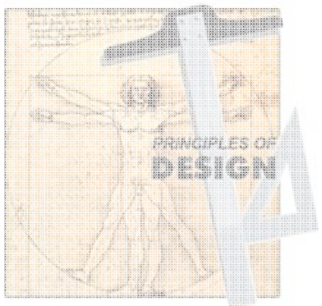
Motivation



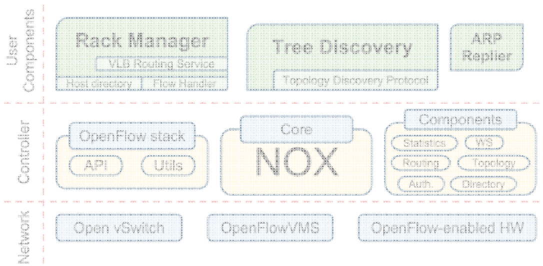
New data center designs

Requirements

SiBF: Switching with in-packet Bloom filters



Design principles



Implementation

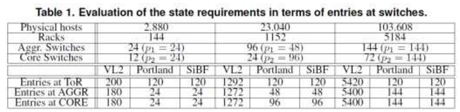


Table 1. Evaluation of the state requirements in terms of entries at switches.

Physical hosts	2.880			23.040			103.608		
Racks	144			1152			5184		
Aggr. Switches	24 ($p_1 = 24$)			96 ($p_1 = 48$)			144 ($p_1 = 144$)		
Core Switches	12 ($p_2 = 24$)			24 ($p_2 = 96$)			72 ($p_2 = 144$)		
	VL2	Portland	SiBF	VL2	Portland	SiBF	VL2	Portland	SiBF
Entries at ToR	200	120	120	1292	120	120	5420	120	120
Entries at AGGR	180	24	24	1272	48	48	5400	144	144
Entries at CORE	180	24	24	1272	96	96	5400	144	144

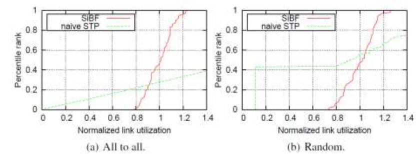


Figure 5. Evaluation of the load balancing behaviour. CDFs of the link utilization.

Evaluation

Future work

Conclusions

State requirements

Assumptions

- ToRs connect 20 servers via 1 Gbps ports and to two AGGRs via 10 Gbps
- 10 concurrent flows per server (5 incoming and 5 outgoing)

Results

Table 1. Evaluation of the state requirements in terms of entries at switches.

Physical hosts	2.880			23.040			103.608		
Racks	144			1152			5184		
Aggr. Switches	24 ($p_1 = 24$)			96 ($p_1 = 48$)			144 ($p_1 = 144$)		
Core Switches	12 ($p_2 = 24$)			24 ($p_2 = 96$)			72 ($p_2 = 144$)		
	VL2	Portland	SiBF	VL2	Portland	SiBF	VL2	Portland	SiBF
Entries at ToR	200	120	120	1292	120	120	5420	120	120
Entries at AGGR	180	24	24	1272	48	48	5400	144	144
Entries at CORE	180	24	24	1272	96	96	5400	144	144

- SiBF and Portland have O(# of ports) vs. VL2 O(# switches) vs. non-scalable vanilla Ethernet O(# of hosts)

Conclusion

- Minimal state at CORE and AGGR (1 entry per neighbour)
- Affordable state at TOR (# simultaneous outgoing flows + # hosted servers)

False positive rate of 96-bit Bloom filters

Setup

- m = 96-bit array
- n = 3 randomly chosen MAC addresses (pool of 1M unique MACs)
- k independent hashes (double hashing with MD5 and SHA-1)
- Tested for 432 (=144*3) randomly chosen MACs
- 10.000 rounds per parameter set

Results

Table 2. Evaluation of the false positive rate of the 96-bit iBF.

k	5	6	7	8	9	10	11	13	15	17	19	21
Theor. Eq 1 ($\cdot 10^{-6}$)	64.89	25.7	11.68	5.95	3.33	2.03	1.32	0.68	0.42	0.31	0.25	0.23
fpr ($\cdot 10^{-4}$)	2.41	1.81	1.5	1.7	1.83	2.23	3.09	4.92	7.17	11.46	16.09	21.07
fpr_{min} ($\cdot 10^{-6}$)	0.93	0.58	1.74	1.85	2.78	5.56	9.72	28.6	95.1	182	355	591

Conclusion

- Deviation from theoretical estimate explained by assumptions [Bose 2008]
- Very low *fpr* suggests few iBF paths with false positives

False-Positive-free forwarding

Setup

- NS-3 implementation
- 3-Tier Clos topo w/48-port AGGRs and COREs (576 ToRs -> 11.520 phy s.)
- Test every combination of ToRsrc - ToRdst (i.e., 331.200 ToR pairs) along each available path (96 typically).
- 30M iBFs sent and accounted for false positives.

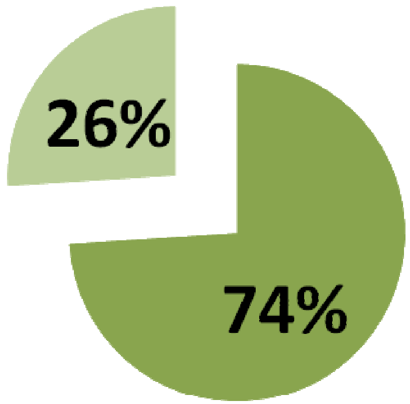
Results

26% of the ToR combinations with some false positive path

- On average, 3 paths (out of 96) with false positives

74% of pairs with every available path false-positive-free

Pairs of ToR_{src}-ToR_{dst}



Conclusion

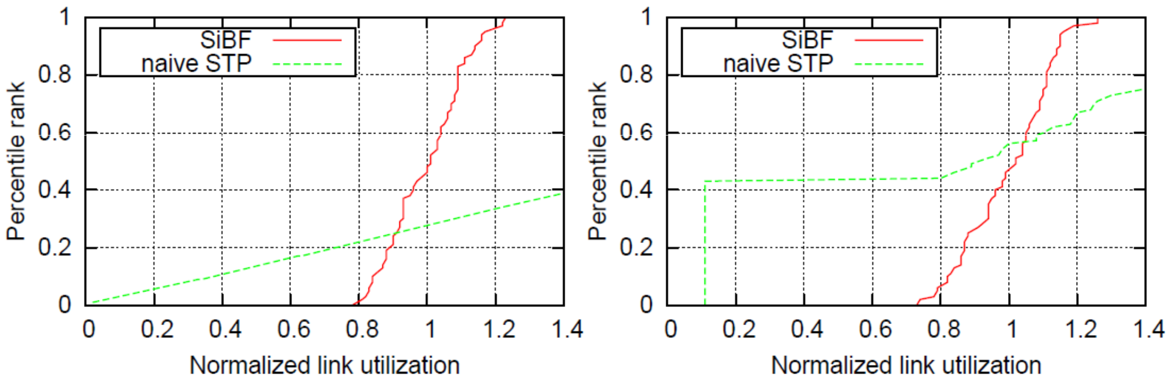
- Only 0.92% of all DCN paths avoided for load balancing
- False-positive-free forwarding comes at an affordable cost (less than 1%) in reduced path multiplicity (can be zeroed w/ d-candidate opt.)

Load Balancing

Setup

- Two synthetic traffic matrices: (1) all-to-all, and (2) random server pairs
- Measure link utilization over 10 rounds
- SiBF Valiant Load Balancing vs. vanilla Ethernet Spanning Tree

Results



(a) All to all. (b) Random.

Conclusions

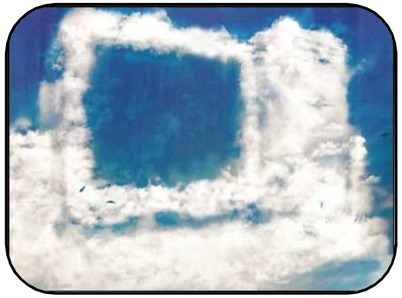
Figure 5. Evaluation of the load balancing behaviour. CDFs of the link utilization.

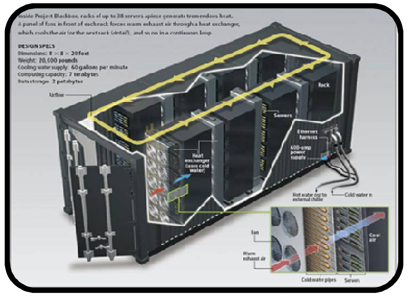
- SiBF splits distributes traffic over every available path reasonable well
- Comparable to other reported VLB implementations (e.g., VL2)
- Better than ECMP (only 16-way +limitations of hash-based flow balancing)

Future Work

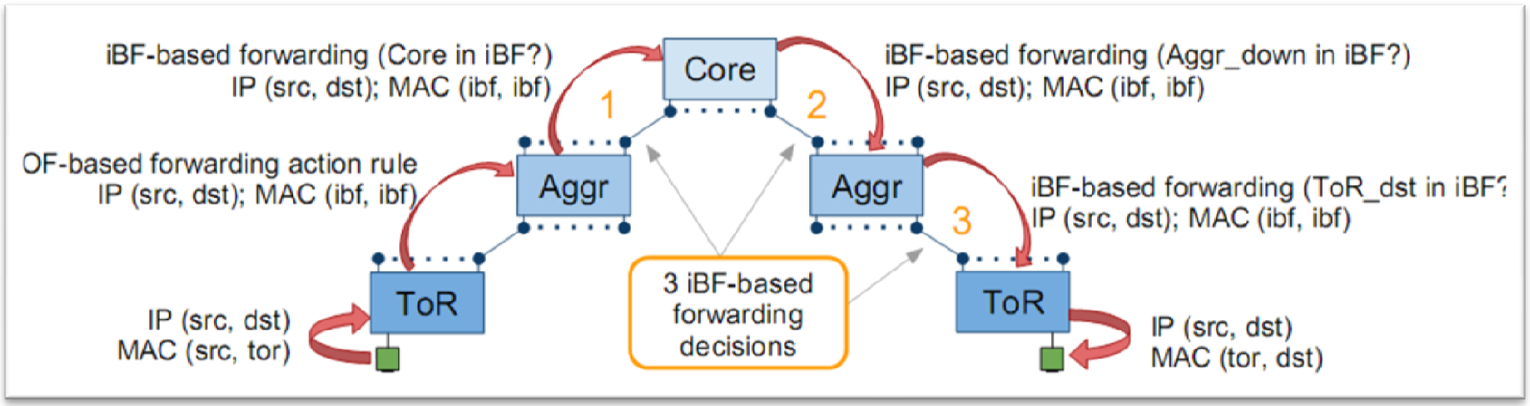
- Flyways for QoS-enabled paths or congestion-free routes via enahanced dynamic load balancing:
  - Re-routing could help avoid losses due to microbursts (requires congestion detection!).
  - MPLS re-route like solution (2^{nd} link-disjoint iBF @ ToR)
- Multicast services
- Seamless workload mobility (VM migration)
- Include middlebox services in the iBF
  - using Bloomed Service Ids or the explicit control path
- Inter-DCN communications (Inter-Cloud VPLS)
- OpenFlow-related (e.g., anycast controllers)

Conclusions

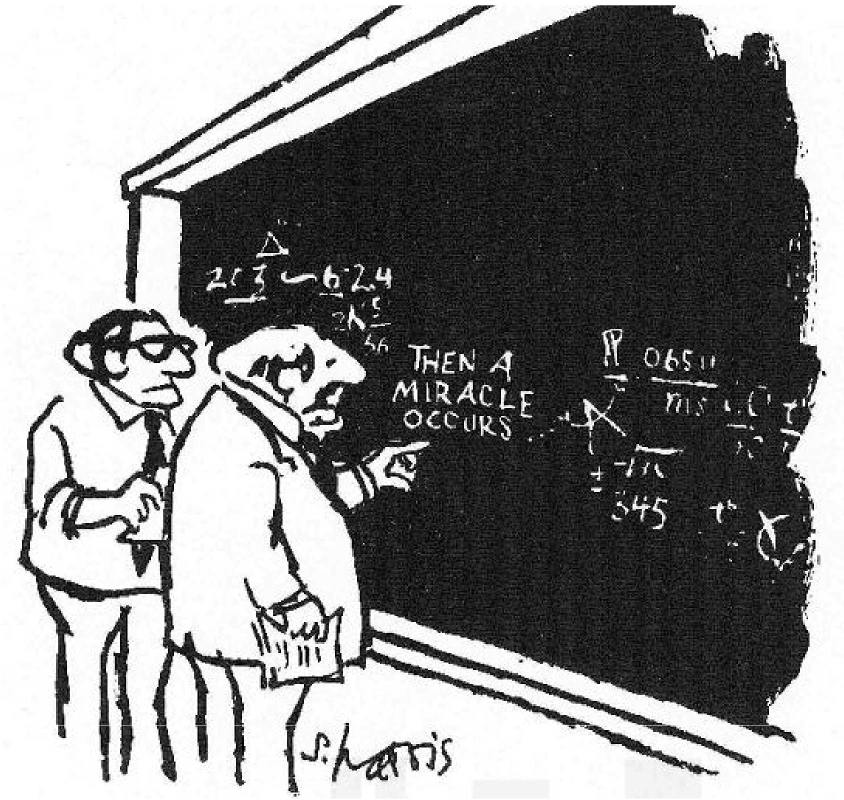




SiBF: Switching with in-packet Bloom filters



SiBF offers *transparent explicit routing, minimal state, load balancing, service differentiation, fault-tolerance, commoditized equipment, etc.*



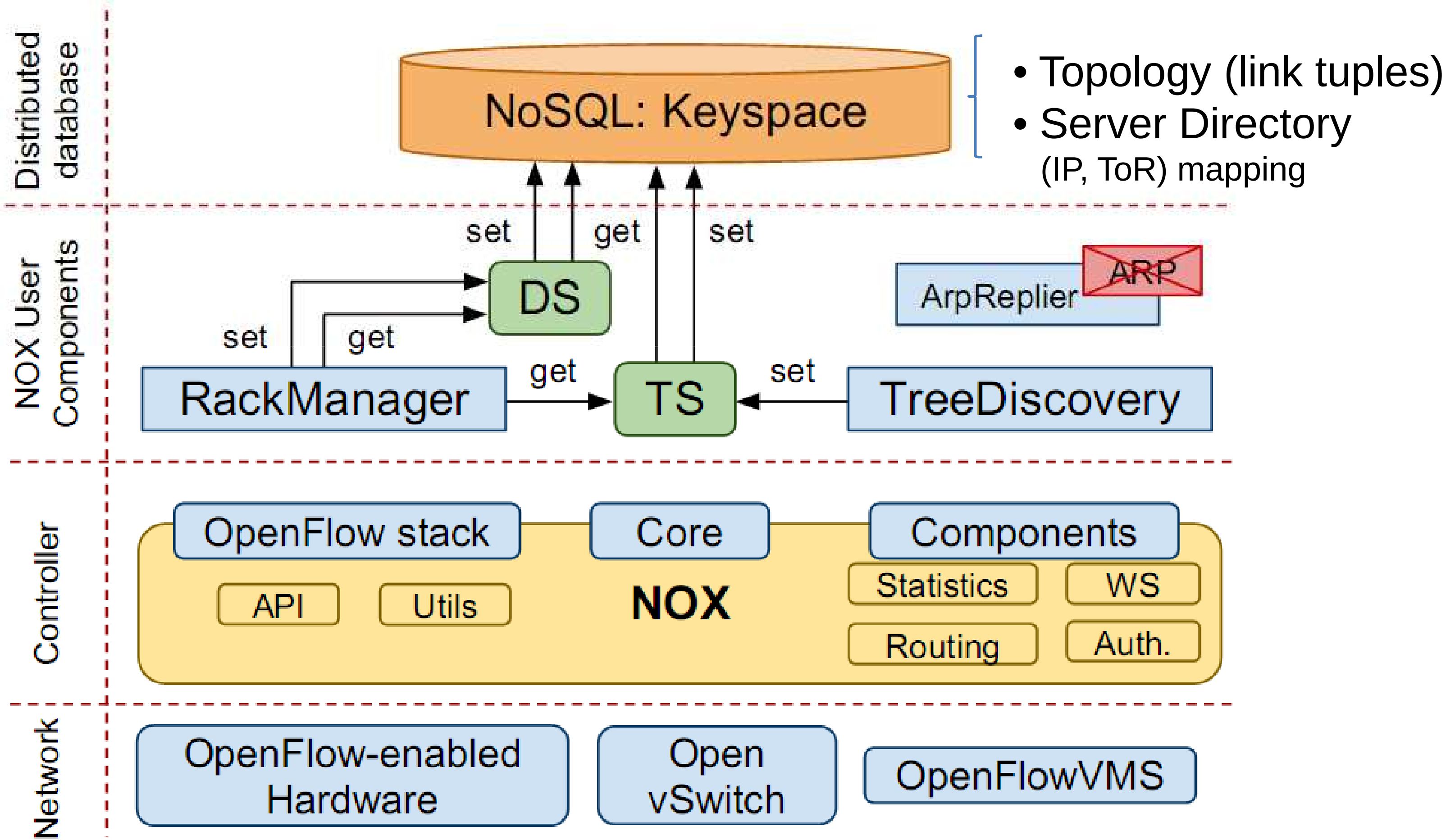
"I think you should be more explicit here in step two"

Thank you!

questions?

BACK-UP

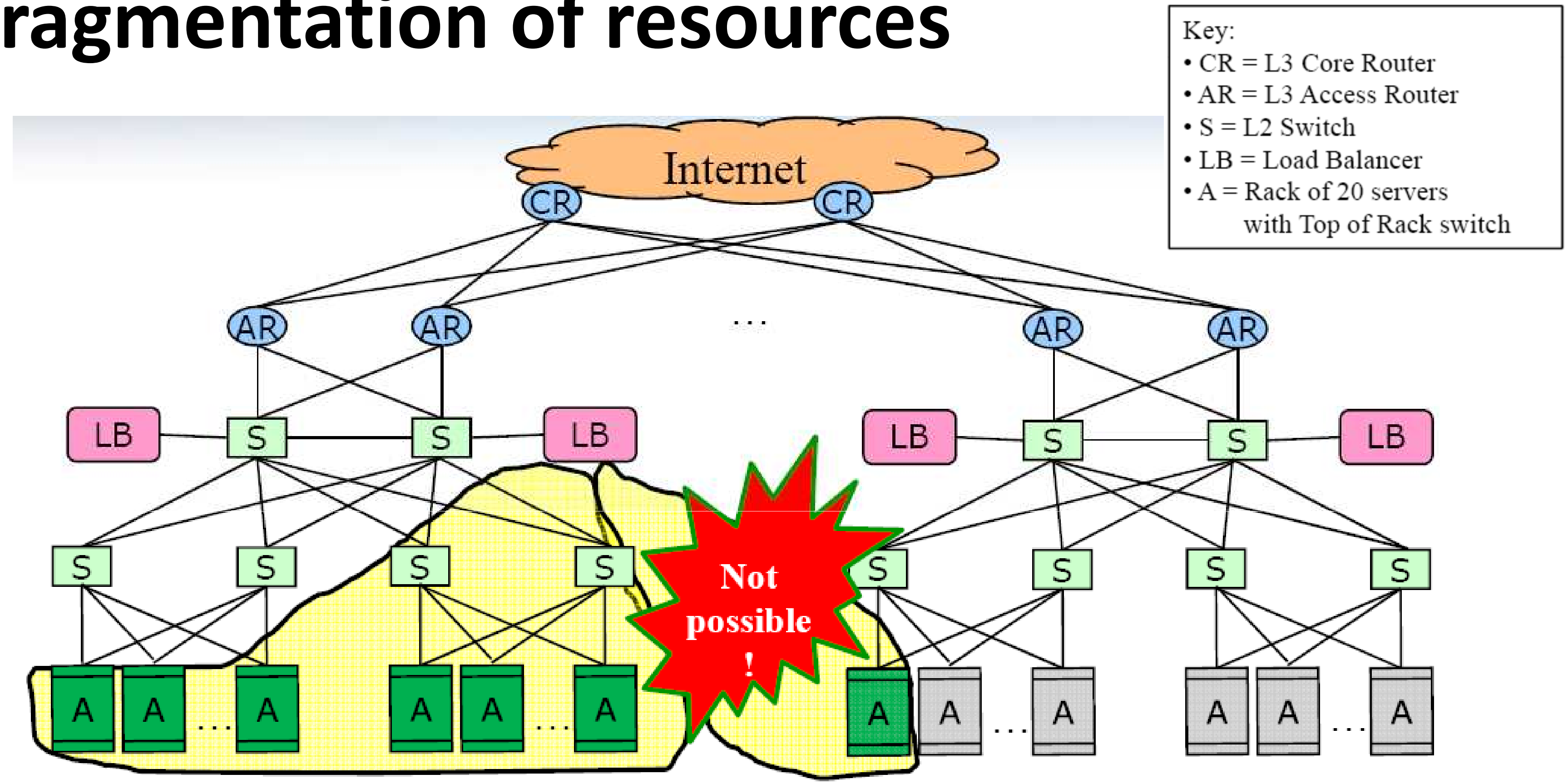
Distributed Rack Manager Architecture



New Generation Data Center Networking

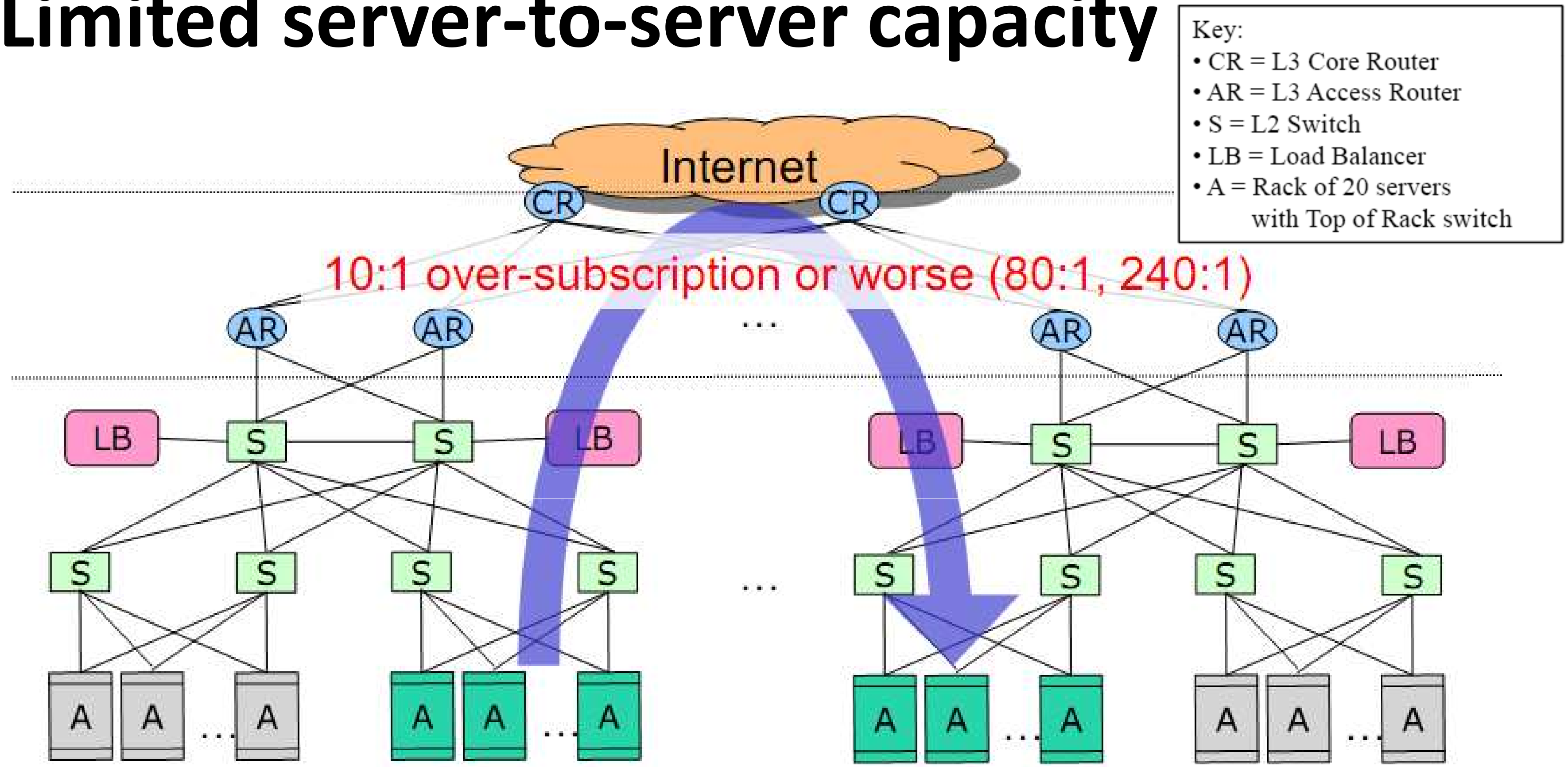
Goals	Requirements	Features
Resource Pooling (servers and network eq.) & Agility	R1: Any VM to any physical machine. - Let services “breathe”: Dynamically expand and contract their footprint as needed - L2 semantics	· ID/loc split · Scalable L2
	R2: High network capacity - Uniform BW and latency for various traffic patterns between any server pair - 1:1, 1:M, N:N efficient communications along any available physical paths	· Multipath support · New TE (load-balancing)
Reliability	R3: Design for failure. - Failures (servers, switches) will be common at scale.	· Fault-tolerance
Low Opex	R4: Low configuration efforts - Ethernet plug-and-play functionality	· Auto-config.
	R5: Energy efficiency - Networking design for idle link/server optimization	· Energy/Cost-awareness
Low Capex	Use commodity hardware	· Scaling-out
Control	Include middlebox services in the data path as required	· Network ctrl.

Fragmentation of resources



– Fragmentation of resources due to load balancers, IP subnets, …
  • limits *agility* to dynamically assign services anywhere in the DC.
– Static Network assignment due to application to VLAN mappings, in-path middleboxes, ...

Limited server-to-server capacity



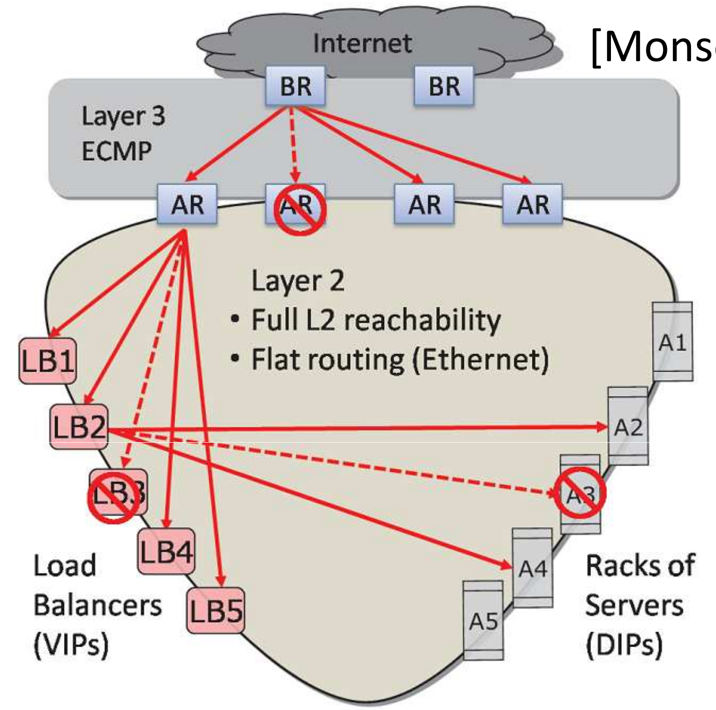
Costly *scale up* strategy to support more nodes and better transfer rates

– Expensive equipment at the upper layer of the hierarchy.

– High over-subscription rates i.e. poor server bisection BW

Layer 2 (Ethernet) scalability

Key:
- CR = L3 Core Router
- AR = L3 Access Router
- S = L2 Switch
- LB = Load Balancer
- A = Rack of 20 servers with Top of Rack switch



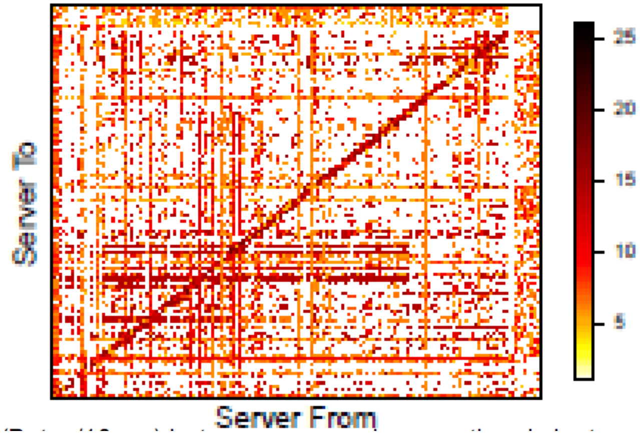
[Monsoon]

x 100.000s

Current layer 2 architectures cannot scale

- limited switch *state* for forwarding tables (flat routing)
- *performance* (bisection BW) limitations (i.e. standard spanning tree protocol limits fault tolerance and multipath forwarding)
- ARP broadcast overhead

DC "traffic engineering"



[IMC09]

- DC traffic is highly dynamic and bursty
  - 1:5 ratio of external vs. internal traffic
  - Traditional traffic engineering does not work well (TM changes constantly)
  - Bursts are too short-lived for traditional approaches to react to them
- Goal of DC traffic engineering
  - Location-independent uniform BW and latency between any two servers
  - For any TM! DC patterns (1:1, 1:M, N:N)
- Approach
  - Avoid spanning tree to make all available paths could be used for traffic
  - Load balancing: E.g., TM oblivious routing, VLB [Monsoon, VLB]
- Additional requirement
  - Force application traffic through middleboxes
    (firewalls, DPI, intrusion det., load balancers, WAN opti., SSL offloaders)

[IMC09] S. Kandula et al. , "The Nature of Data Center Traffic: Measurements and Analysis", To Appear in IMC 2009