Fog and Cloud Computing 2025/26 - Autumn 2025 - UiO

Load Balancing and Cyber-foraging

Prof. Paulo Ferreira

paulofe@ifi.uio.no

UiO/IFI/PT

Load Balancing in Data Center Networks: A Survey. J. Zhang, F. R. Yu, S. Wang, T. Huang, Z. Liu and Y. Liu. *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2324-2352, 2018, doi: 10.1109/COMST.2018.2816042.

© Paulo Ferreira

Introduction to Load Balancing

© Paulo Ferreira

Introduction to Load Balancing (1/2)

- In recent years, enterprises and corporations are g**radually shifting their services** like Web search and online gaming **into cloud environments**
- To support these ever increasing cloud services, a **large number of data centers have been deployed**
- Today's data centers usually employ the **scale-out model, such as FatTree, VL2, DCell,** to connect a **large number of commodity servers** and thus support large scale computation
- In these data center architectures, it is common that **multiple paths exist between a pair of end-hosts**
- Therefore, **it is important to evenly balance traffic load** across multiple paths

© Paulo Ferreira

Introduction to Load Balancing (2/2)

- However, the **traffic patterns in data center networks are quite different from that in traditional Internet**

- First:
  - **big-data computation frameworks** like MapReduce and Dyrad exhibit **partition/aggregate pattern**, where a central server assigns jobs to distributed servers to process and then aggregates results from them
  - this **many-to-one communication pattern** possibly causes **throughput collapse**
- Second:
  - the flow arrival processes of some applications follow **ON-OFF patterns**, which increases the **burstisness and unpredictability** of the traffic matrix
- Third:
  - most flows in data centers are **short in size**, like 10 KB
  - but there are a **small amount of large flows** that contribute to most bytes of the traffic and likely lead to traffic congestion

© Paulo Ferreira

Problems with Traditional Load Balancing

- Due to the special topology and traffic characteristics, **traditional load balancing algorithms are not well suited for the data center environment**
- On one hand:
  - load balancing schemes for traditional networks are designed in large scheduling periods and thus **fail to well balance bursty traffic in data centers**
- On the other hand:
  - **conventional static load balancing algorithms are agnostic to congestion information**, which may lead to congestion in some scenarios
- For example:
  - the widely deployed ECMP ("equal-cost multi-path routing") employs a **static hashing mechanism** according to several fields in packet headers, such as source IP and destination IP, and then maps the hash value to one of the equal-cost paths to the next hop
  - ECMP is **convenient to be deployed in existing networks**; however, it is **easy to cause congestion due to large flows collisions in data center networks**

© Paulo Ferreira

Challenges of Load Balancing (1/2)

- Besides the topology and traffic, the design of load balancing mechanism faces several **new challenges in data center networks**

- The first challenge:
  - how to **timely and accurately detect congestion** and **measure the bottleneck bandwidth** of the path
  - since the data center **traffic is very bursty and the RTT (Round-Trip-Time) of the network is at microseconds level**, the **load balancing has to react to congestion at RTT timescales** (∼ 100 μs)

- Second:
  - load balancing should be **robust against reordering**
  - **packets from the same flow could be balanced to different network paths**, which possibly causes TCP (Transmission Control Protocol) reordering
  - **reordering** not only **impacts the throughput of TCP**, but also **imposes significant computational overheads on hosts**

© Paulo Ferreira

Challenges of Load Balancing (2/2)

- Third:
  - load balancing **indiscriminately applies the same set algorithms to short and long flows**
  - however, **short flows are more latency sensitive than long flows**, thus the two types of flows need to be handled very differently
- Fourth:
  - **link failures have been shown to be frequent and disruptive in data centers**, thus load balancing has to effectively **handle asymmetry caused by link failures**
- Finally:
  - load balancing faces many minor **deployment challenges** in data centers
  - for **example**, how to implement new load balancing algorithms in hardware switches, how to sense all routing paths at end hosts, how to explicitly control the routing path of a flow at end-hosts

© Paulo Ferreira

Summary of Load Balancing

- Therefore, during the last several years:
  - researchers and engineers have **proposed a large number of mechanisms to address the load balancing problem in data centers**
- The focus of these mechanisms varies a lot
- In summary:
  - the objectives of a load balancing mechanism mainly include **throughput, latency, robustness, scalability, and energy efficiency**
- Specifically:
  - 1) **Throughput** and **latency** focus on improving network utilization and reducing flow completion time
  - 2) **Robustness** refers to how the mechanisms react to topology changes such as link failures and switch crashes
  - 3) **Scalability** requires that the mechanisms could be deployed in large scale data center networks with acceptable cost
  - 4) **Energy-efficiency mechanisms** try to finish jobs with a modest number of devices to reduce energy consumption

© Paulo Ferreira

Background-Link Load Balancing vs. Server Load Balancing (1/2)

- Load balancing mechanisms summarized in this survey aim to balance traffic across links, which can be called **link load balancing** (or **network load balancing**)
- To avoid confusion with another widely used term, **server load balancing**, next we briefly introduce the concept and some mechanisms of server load balancing
- **Server load balancing mechanisms are designed to balance traffic between servers**
- They can be further classified into:
  - **layer 4** and **layer 7** server load balancing mechanisms
- **Layer 4 server load balancing mechanisms are unaware of application information**:
  - they leverage **IP address and port information** to determine where traffic will be directed to
  - layer 4 server load balancing plays an important role in scaling online Internet services, such as Web search, e-commerce
  - these online services usually run on multiple servers in data centers, and each server has an **individual direct IP (DIP)**
  - a service exposes one or more **virtual IP addresses (VIP)** to users
  - the load balancer receives the traffic destined for the VIP, and balances it to a corresponding DIP

© Paulo Ferreira

Background-Link Load Balancing vs. Server Load Balancing (2/2)

- **Layer 7 server load balancing mechanisms are aware of application information**:
  - they **inspect the content of requests and route requests** based on the information of application layer
- **For example**, layer 7 server load balancing can **balance all requests for text files across servers A and B, and balance all requests for graphics files across servers C and D**:
  - besides, **one of the main tasks of layer 7 server load balancing is to maintain persistence** (guarantee the requests from a client are routed to the same server)
  - furthermore, in some scenarios, **layer 7 server load balancing can rewrite requests and replies to hide internal URLs**
- Thus, **compared to link load balancing introduced in this survey, layer 4 and layer 7 server load balancing mechanisms mainly distribute requests to different servers based on the load and content of servers**
- **While link load mechanisms mainly balance traffic across different links to avoid congestion of links**

© Paulo Ferreira

Load Balancing

© Paulo Ferreira

What is Load Balancing?

https://www.youtube.com/watch?v=y-4zEpfQ-sw

© Paulo Ferreira

Background to Load Balancing

- In recent years:
  - **data center networking has attracted much attention** in both academia and industry
- There are some existing survey literatures about data center networks:
  - such as **transport control, bandwidth allocation, data center virtualization**
- Since the design of data center **load balancing mechanisms** needs to mainly consider **data center topology and traffic characteristics**:
  - we provide some essential **background knowledge on the architectures of data center networks** and the **traffic characteristics** of data center applications
  - furthermore, some data center load balancing mechanisms take **energy consumption as an important metric**
  - thus, we also present some **background information on green data centers**

© Paulo Ferreira

Background-Data Center Network Architecture (1/10)

- In order to provide **large bandwidth**, **reduce the cost of large-scale deployment**, and **enable performance isolation of different applications**:
  - several **network architectures** have been proposed for data center networks.
- These **architectures** could be classified into:
  - **switch-centric**,
  - **server-centric**, and
  - **hybrid structures**
- In switch-centric architectures:
  - **switches are used to provide multiple paths between servers and perform packet forwarding**
- In server-centric architectures:
  - **servers both perform computation functions and act as switches to connect to other servers**
- Hybrid architectures:
  - **uses both switches and servers** to perform packet forwarding and usually provides multiple paths with unequal length between servers

© Paulo Ferreira

Background-Data Center Network Architecture (2/10)

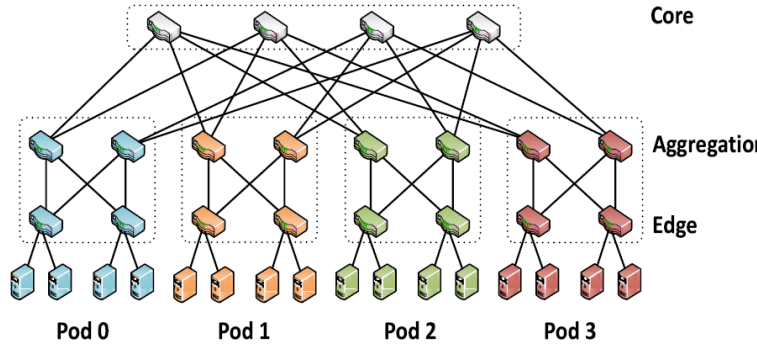
- ***Fat-Tree:***
  - Fat-tree is a layered **switch-centric architecture**
  - it aims to maximize end-to-end bandwidth with high scalability

- A fat-tree topology has three layers named:
  - **edge**,
  - **aggregation**, and
  - **core**

- In a ***k*-ary fat-tree**, there are ***k* pods**:
  - **each pod contains two layers of *k*/2 switches**
  - **each switch has two layers of *K* ports**
  - **in the edge layer, *k*/2 ports of a switch connect to *k*/2 hosts and the remaining *k*/2 ports of the switch are connected to the switches in the aggregation layer**
  - **in the core layer, there are $k^2/4$ core switches and each of them has one port connected to one of the *k* pods**

© Paulo Ferreira

Background-Data Center Network Architecture (3/10)



- In general, there are **$k^3/4$ hosts, $k^2/2$ edge switches, $k^2/2$ aggregation switches, and $k^2/4$ core switches in a *k*-ary fat-tree architecture**
- The Fig. in this slide shows a **fat-tree network topology where *k* = 4**
- Fat-tree **greatly reduces the equipment cost by employing commodity switches**, but **incurs higher cabling cost**

© Paulo Ferreira

Background-Data Center Network Architecture (4/10)

- ***VL2****:*
  - VL2 is also a **Clos-based switch-centric architecture and is divided into edge layer, aggregation layer, and core layer**
- In the architecture:
  - VL2 uses 1GbE links to connect hosts and edge switches, and
  - 10GbE links for upper layer switch-switch interconnection
- Assuming that edge switches have *n*0 10GbE ports and 10*n*0 1GbE ports (E means Ethernet):
  - aggregation switches have *n*1 10GbE ports, and core switches have *n*2 10GbE ports
  - each switch in the edge layer is connected to 10*n*0 hosts via 1GbE ports and *n*1 aggregation switches via 10GbE ports
  - each switch in the aggregation layer uses *n*1/2 10GbE links to connect to edge switches and the rest links to connect to core switches
  - we could calculate that there are totally *n*1/2 core switches, *n*2 aggregation switches, and *n*1**n*2/2*n*0 hosts in the described VL2 architecture

© Paulo Ferreira

Clos Network (1/3)

- Origin:
  - Charles Clos was a researcher at Bell Laboratories in the 1950s
  - he published a paper titled "A Study of Non-blocking Switching Networks" in the Bell System Technical Journal in 1953
  - he described how telephone calls could be switched with equipment that used multiple stages of interconnection to allow the calls to be completed
  - the switching points in the topology are called crossbar switches
  - Clos networks were designed to be a three-stage architecture, an ingress stage, a middle stage, and an egress stage
  - the concept is that there are multiple paths for the call to be switched through the network so that calls will always be connected and not "blocked" by another call
  - the term fabric came about later because the pattern of links looks like threads in a woven piece of cloth
- E.g.:
  - https://www.networkworld.com/article/2226122/clos-networks-what-s-old-is-new-again.html

© Paulo Ferreira

Clos Network (2/3)

- Clos Network within Network Switches:
  - first created in the mid-1950s as a method to switch telephone calls
  - Clos networks made a reappearance many years later in the 1990s when early Ethernet switches were being developed
  - to create connectivity where any Ethernet interface on a switch could send Ethernet frames to any other interface on that switch, there needed to be a similar crossbar matrix of connectivity within the switch
  - the number of interfaces in the switch governed how large the crossbar fabric needed to be; when modular chassis-based network switches were developed, the crossbar switching fabric needed to grow to accommodate faster interface speeds; the crossbar fabric was provided by the supervisor module combined with the wiring between cards within the chassis
  - over time, Ethernet switches were developed that had input and output queues on all the interfaces
  - modern Ethernet switches have more advanced fabric technologies, output queuing and priority-based flow control so they can now achieve non-blocking performance
  - with these technical enhancements, switches can now support guaranteed bandwidth connectivity for protocols using 10 Gigabit Ethernet links

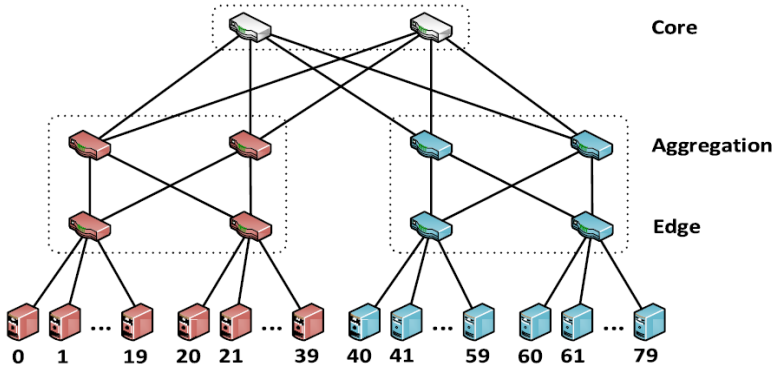
© Paulo Ferreira

Clos Network (3/3)

- Data Center Switching Using Clos Architecture:
  - networks started to use the "fat tree" model of connectivity using the core - distribution - access architecture
  - the link speeds got progressively higher as you reached the core
  - for example, the access links to servers or desktops might have historically been 100Mbps Fast Ethernet links, the uplinks to the distribution switches might have been 1Gbps Ethernet links, and the uplinks from there to the core would have been 4X1Gbps port channels
  - the problem with traditional networks built using the spanning-tree protocol or layer-3 routed core networks is that a single "best path" is chosen from a set of alternative paths
  - all data traffic takes that "best path" until the point that it gets congested then packets are dropped
  - the alternative paths are not utilized because the topology algorithm deemed them to be less desirable or removed to prevent loops from forming
  - there is a desire to migrate away from using spanning-tree while still maintaining a loop-free topology yet utilizing all the multiple redundant links

© Paulo Ferreira

Background-Data Center Network Architecture (5/10)

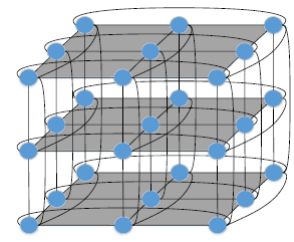


- The Fig in this slide gives a **VL2 topology** where *n*0=2, *n*1=4, *n*2=4
- The **main advantage of VL2** is the ability to provide high bandwidth (i.e., bandwidth between two partitions) with commodity switches

© Paulo Ferreira

Background-Data Center Network Architecture (6/10)

- *CamCube:*
  - **CamCube is a server-centric architecture** that uses only servers to connect with others in a 3D-Torus manner
- The Fig in this slide shows a **CamCube architecture** with 27 servers
- The **server-centric architecture could significantly reduce the cost of switches and routers**
- It is also **energy efficient** since servers need fewer cooling cost than switches
- The **main disadvantage of CamCube** lies in that paths may become quite long for applications, and the routing complexity may be high



Blue circles are servers

© Paulo Ferreira

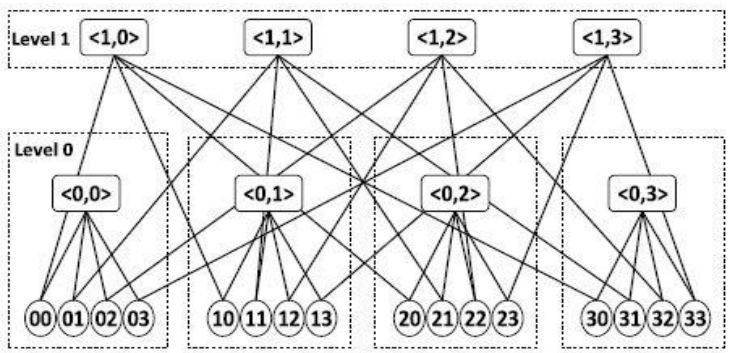
Background-Data Center Network Architecture (7/10)

- ***Bcube* is a hybrid architecture** that uses both switches and servers with multiple ports as forwarding devices:
  - it could be defined recursively
  - a *BCube*0 consists of *n* servers connected to a *n*-port switch;
  - a *BCube*1 is composed of *n* *BCube*0s and *n* *n*-port switches
  - recursively, a $Bcube_k$ *(*$k \geq 1$*)* contains *n* $Bcube_{k-1}$s and n^k *n*-port switches
- There are totally n^{k+1} servers with *(*$k + 1$*)* ports and $k + 1$ levels of switches
- To construct a $Bcube_k$, we:
  - first number the *n* $Bcube_{k-1}$s from 0 to $n - 1$, and the n^k servers in each $Bcube_{k-1}$ from 0 to $n^k - 1$;
  - then we connect the level-*k* port of the *i*-th server in the *j*-th $Bcube_{k-1}$ to the *j*-th port of the *i*-th level-*k* switch where $i \in [0, n^k - 1]$ and $j \in [0, n - 1]$

© Paulo Ferreira

Background-Data Center Network Architecture (8/10)

- The Fig. in this slide presents a ***BCube* with *n* = 4**
- **BCube can provide high bandwidth with low cost**, but a significant **disadvantage** of BCube is that its **recursion depth is limited by the NIC (Network Interface Card) number of servers**



Boxes are switches, and circles are servers

© Paulo Ferreira

Background-Data Center Network Architecture (9/10)

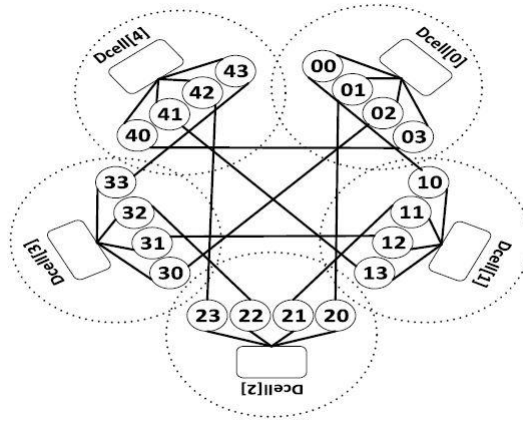
- ***Dcell* is another hybrid architecture** where a server is connected to a mini-switch and several other servers:
  - it can also be defined recursively
  - a *DCell*0 is constructed from *n* servers and a mini-switch
  - each server connects to the mini-switch via a 1GbE or 10GbE link
- In the prototype of Dcell:
  - *n* is a small integer (usually $n \leq 8$), so a commodity 8-port switch is enough for the architecture
  - level-1 *DCell*1 consists of *n* + 1 *DCell*0s
  - we mark each server with a 2-tuple [*d*1*, d*0], where *d*1 and *d*0 represent the IDs of level-1 and level-0 respectively
  - then we connect server pairs marked with [*i, j* − 1] and [*j, i*] for every *i* if *j* > *I*
  - if we treat each *DCell*0 as a single node, then the *DCell*1 becomes a complete graph
  - higher levels of DCell can be constructed in the same way

© Paulo Ferreira

Background-Data Center Network Architecture (10/10)

- The Fig. in this slide depicts a *DCell* architecture with *n* = 4
- The main advantage of DCell is that it provides large bandwidth using commodity switches and high scalability:
  - however, paths between servers could be quite long in large-scale topology

Boxes are switches, and circles are servers.



© Paulo Ferreira

Other Architectures

- Other architectures that are not presented in this survey are:
  - Switch-centric: JellyFish [41], PortLand [42]
  - Server-centric: Hyper-BCube [43], MDCube [44],
  - FiConn [45], FlatNet [46], SprintNet [47]
  - Hybrid structures: OSA [48], Helios [49], c-Through [50]

© Paulo Ferreira

Background-Data Center Traffic Characteristics

© Paulo Ferreira

Background-Data Center Traffic Characteristics (1/5)

- There are **different applications running in data center networks** ranging from **latency-sensitive** services like Web search, to **throughput-sensitive** tasks like database update
- With the growing applicability of data centers in a wide variety of scenarios, there have been **several systematic measurement studies** of data center usage to guide practical issues in data center operations
- We **show three key results for data center network design** from these studies:
  - first, **traffic patterns in a data center are various and very dynamic**, but there are still some characteristics about flow numbers and total bytes
  - second**, link utilization is higher in top layers in a data center**; however, **congestion likely happens at bottom layers** since congestion is more related to traffic bursts instead of link utilization
  - lastly, although there are backup equipments and redundant paths in data centers to improve reliability, **failures still happen sometimes and result in significant downtimes**
- We now analyze these characteristics in detail

© Paulo Ferreira

Background-Data Center Traffic Characteristics (2/5)

- *1)* ***Data Center Traffic Analysis*** from the analysis of Netflow and SNMP data, several **macroscopic characteristics of data center traffic** could be observed:
  - first, the **destination of traffic in data center varies with owners**
  - nearly **80% traffic in cloud data centers stays within the rack**, while **40%-90% traffic traverses across different racks in university and private enterprise data centers**
  - second, the **majority of flows in data centers are small in size (a few KB),** which are mostly contributed by requests to distributed file systems; there are only a small amount of flows whose length vary from 100 MB to 1 GB, and the **number of flows over a few GB is very small**
  - third, **although most flows are small in size in a data center, the total bytes are mainly contributed by the small fraction of large flows**

© Paulo Ferreira

Background-Data Center Traffic Characteristics (3/5)

- ***2) Data Center Congestion Analysis:***
  - it is intuitive to think that **loss rates are positively associated with link utilization**,
  - **but recent studies show that they are irrelative and some special traffic pattern which cause severe packets loss**

- Besides, **link utilization grows rapidly in higher layers**:
  - for example, **links in core layers are often observed to achieve as high as 70% link utilization**
  - however, **core links surprisingly exhibit the smallest loss rates** despite their high link utilizations
  - the research suggests that these **packet losses are mainly caused by microbursts** in each layer and **traffic is more bursty in aggregation and edge layers than in core layer**

- Another observation is that **a small fraction of links experience more packet losses than others due to congestion**:
  - this could be alleviated by employing proper load balancing mechanisms

© Paulo Ferreira

Background-Data Center Traffic Characteristics (4/5)

- ***3) Failures in Data Centers:***
  - data centers usually provide high reliability by setting **backup equipment and redundant paths**
  - but **link failures and device failures still happen** at times in data centers

- The reliability and survivability of data center networks are related to network topologies:
  - it is **desirable for load balancing algorithms to promptly react** to network failures

- There are four key observations about failures:
  - first, **low-cost commodity switches are highly reliable with only a small failure rate**
  - second, **software faults could lead to both device failure and link failure**
  - third, **failures will cause a large number of small packets dropping, which greatly degrades TCP performance**
  - lastly, **network redundancy helps to reduce the effect of failures**

© Paulo Ferreira

Background-Data Center Traffic Characteristics (5/5)

- **Current approaches to balance load in data centers are mostly borrowed from mature techniques in traditional networks**, such as ECMP
- However, **they are designed for relatively stable traffic in traditional networks**
- A **well-designed load balancing mechanism for data center networks should take the previously characteristics observed into consideration**
- More specifically:
  - **the mechanism requires to route and re-route large flows in microsecond level,**
  - **transmit small flows even faster, and**
  - also needs to **promptly react to failures and re-route flows away from congestion-related links**

© Paulo Ferreira

Background-Data Center Energy Consumption (1/5)

- The **number and size of data centers are growing exponentially:**
  - to accommodate increasing demands of users and applications
- Meanwhile, the **energy needs and electricity costs of data centers also draw growing concerns**
- As a result, the **network infrastructure of data centers consumed approximately 15.6 billion kWh of energy in 2010**, and the energy consumption continued to grow
- **Network components, servers, and cooling systems** are:
  - the **three major consumers in data centers**, and **network components contribute much more consumption than the other two**
  - therefore, **researchers are trying to design energy efficient network infrastructure to build green data centers**
- The three major directions to achieve this goal are:
  - **energy-efficient architecture**,
  - **resource allocation**, and
  - **traffic management**

© Paulo Ferreira

Background-Data Center Energy Consumption (2/5)

- *1) **Energy-Efficient Architecture**:*
  - **traditional three-tier architecture used in data centers** usually **leverages power hungry enterprise-class network equipment at higher layers of the network**, which contributes to most of the energy consumption of network components
- Therefore, an optional method is to **replace power-hungry enterprise-class equipment in legacy data centers with energy-efficient commodity equipmments**
- Some architectures, such as **Fat-Tree and VL2**, could achieve better bandwidth with commodity equipmments:
  - e.g., a Fat-Tree data center **consumes around 56.6% less energy and dissipates around 56.5% less heat**

© Paulo Ferreira

Background-Data Center Energy Consumption (3/6)

- **Another method to achieve energy-efficient architecture is workload consolidation**:
  - on average, **data center network remains highly underutilized with an average load of around 5%-25%**
  - thus, **some architectures, such as ElasticTree** (dynamically adjusts the set of active network elements — links and switches — to satisfy changing data center traffic loads, similar to FatTree)**, leverage this feature to consolidate the workload on a subset of network resources to save energy**
  - the **ElasticTree saves around 50%** energy compared to traditional architectures

© Paulo Ferreira

Background-Data Center Energy Consumption (4/6)

- ***2) Resource Allocation:***
  - servers are also major contributors to energy consumption in data centers, and a **larger number of running servers result in higher power consumption**
  - therefore, **optimizing the utilization of servers** can lead to more energy savings in data center networks
- Virtual Machine (VM) placement is a hot-topic area in data centers in these years:
  - by **properly placing VMs according to the network capability and resource requirements of applications**, the network manager can save unnecessary resource for other services and **make the data center network more energy-efficient**
- Some theoretical server virtualization approaches have been applied in production environments recently:
  - it leads to **massive savings in the power consumption of hardware and cooling systems in data centers**

© Paulo Ferreira

Background-Data Center Energy Consumption (5/6)

- ***3) Traffic Management:***
  - **data center networks typically hold an average network load of less than 25% of the peak load**, and a **considerable amount of links in data center networks remain idle for 70% of the time**
  - by **redirecting the network traffic across different links, the network manager can put more routers to idle state and place them in sleep mode**
  - this method is usually used in energy-efficient architectures to **transmit traffic with a subset of the network components while satisfying performance such as link utilization and packet delay**
  - however, **traffic management can only improve energy efficiency in a certain level**, because there is always a **trade-off between service requirements and energy consumption**
- One important issue in data center traffic management:
  - **load balancing aims at reducing packet delay**, but
  - **it can also redirect flows by considering energy-efficient targets**

© Paulo Ferreira

Background-Data Center Energy Consumption (6/6)

- Based on the **specific characteristics of topology and traffic**:
  - a **load balancing mechanism is urgently needed to schedule flows in data centers**
- Furthermore:
  - with the growing concern of energy consumption and green data centers, more and more researchers are trying to **improve energy efficiency through scheduling load in data centers**

© Paulo Ferreira

Understanding Load Balancing

© Paulo Ferreira

Understanding Load Balancing

- As **today's data centers provide high path diversities between hosts and exhibit special traffic characteristics**:
  - a proper load balancing mechanism is required to **improve application performance and network utilization**
- We now introduce the **load balancing problem in data center networks**:
  - from its definition, features, and objectives to help readers quickly understand the problem in data center networks

© Paulo Ferreira

Understanding Load Balancing-Definition of Load Balancing (1/3)

- The problem of scheduling flows through a capacitated network simultaneously is featured as **multi-commodity flow (MCF) problems**, which has already been studied practically and theoretically
- The traffic scheduling problem in data centers is to **find the optimal flow assignments for the following maximum MCF problem**:
  - The target function *Ui(xi)* represents per commodity utility, which depends on the sending rates of flows
  - *xi* is the sending rate of flow *i*
  - $f^{s,d}_{u,v}$ is the flow rate across links *(u, v)* between adjacent switches
  - the *(s, d)* pair represents the source and destination of the flow

$$\text{maximize:} \quad \sum_i U_i(x_i)$$

$$\text{subject to:} \quad \sum_{u:(s,u)\in E} f^{s,d}_{u,v} = \sum_{w:(v,w)} f^{s,d}_{v,w} : \forall v, s, d,$$

$$\sum_{u:(s,u)\in E} f^{s,d}_{s,u} = \sum_{i:s\to d} x_i : \forall s, d$$

$$\sum_{s,d} f^{s,d}_{u,v} \le c_{u,v} : \forall (u, v) \in E, \text{ link capacity } c_{u,v}$$

© Paulo Ferreira

Understanding Load Balancing-Definition of Load Balancing (2/3)

- The **load balancing algorithms are targeted on flow assignments across links**
- **Load balancing alone could not guarantee an optimal solution to the MCF problem since the optimal solution also relies on sending rates of flows**
- Generally, **the sending rates are decided by transport control protocols like TCP at hosts**:
  - in a symmetric network with fixed sending rates *xi*, the load balancing algorithm is equivalent to minimizing the maximum link utilization

$$min\ max_{(u,v)\in E} \frac{\sum_{s,d} f_{u,v}^{s,d}}{c_{u,v}}$$

- If this problem can be solved, **the solution can be leveraged to determine the optimal scheduling policy**:
  - however, the **MCF problem is NP-hard**, and it is very challenging to solve the problem in microsecond level
  - thus, the **load balancing mechanisms generally propose new heuristic algorithms to solve the problem**

© Paulo Ferreira

Understanding Load Balancing-Objectives of Load Balancing (3/3)

- The **main target of load balancing algorithms in data centers is to evenly assign flows among different paths to improve network utilization**
- The **scheduling granularity** could be **a flowlet, or a flowcell with fixed length:**
  - a **flowlet** is a burst of packets that is separated in time from other bursts in one TCP flow by an idle time interval that is larger than a predefined timeout value
  - a **flowcell** is a near-uniform units of data that fit within end-host segment offload optimizations used to support fast networking speeds
- Since the **applications in data centers have their specific requirements**:
  - the load **balancing algorithms should take this into account** and sometimes need to make a **tradeoff between different requirements**
- There are four primary objectives of the load balancing algorithm:
  - **high performance**,
  - **scalability**,
  - **robustness**, and
  - **energy-efficiency**

© Paulo Ferreira

Comparison of Traffic Scheduling in Wide Area Networks and Data Centers

© Paulo Ferreira

Comparison of Traffic Scheduling in Wide Area Networks and Data Centers

- We first describe **the traffic scheduling problem and several typical solutions in WANs**
- Then, **the differences of traffic scheduling between WANs and data centers** are analyzed
- With the **rapid growth of services and user demands**:
  - **provisioning an ISP backbone to satisfy the requirements of different services** is becoming ever more challenging

© Paulo Ferreira

Comparison of Traffic Scheduling in Wide Area Networks and Data Centers-Traffic Engineering/Scheduling in WANs

- Compared to data center networks, **WANs have properties of long-distance transmission**, **limited bandwidth resource, expensive hardware** and so on:
  - these properties make **congestion happen frequently at some links while leaving other links under-utilized**
  - thus, **a traffic scheduling mechanism is required to balance load and meet the Service Level Agreements** (SLAs) of customers
- The main scheduling manner in WANs is to **differentiate traffic of services** and then **assign different amount of resources to them**
- Three representative traffic scheduling solutions proposed for WANs:
  - **IntServ**,
  - **DiffServ**, and
  - **MPLS TE**

© Paulo Ferreira

Comparison of Traffic Scheduling in Wide Area Networks and Data Centers-Differences Between WANs and Data Centers (1/3)

- There are several differences between **traffic scheduling** in wide area networks and in data centers

- **1-Network topologies:**
  - **WANs are typically irregular**, which may have bottlenecks themselves
  - reversely, **data center topologies are usually symmetric or regular with several equal-cost paths between each pair of end-hosts**, which provides high bandwidth and reduces the complexity of computing link costs

- **2-Link capacities and hardware types vary a lot in WANs**:
  - thus, a **traffic scheduling mechanism requires to take different limitations of nodes into account when assigning traffic**
  - in contrast, **in one data center layer, switches usually belong to the same type, and link capacities are also identical, which greatly simplifies the traffic scheduling problem**

© Paulo Ferreira

Comparison of Traffic Scheduling in Wide Area Networks and Data Centers-Differences Between WANs and Data Centers (2/3)

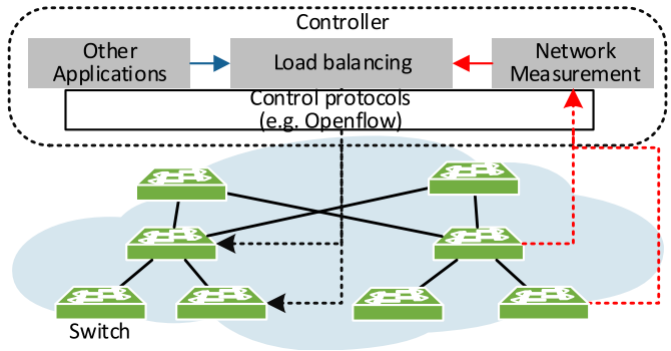
- **3-Service changes and scheduling adjustments usually take hours to days or even longer in WANs**:
  - by contrast, **in a data center network with high-speed links**, short flows only need **tens to hundreds of milliseconds to finish**
  - **congestion caused by bursts also disappears in millisecond level**, which requires the **traffic scheduling algorithm to react promptly enough to process them**

- **4-WANs typically do not have centralized controller**:
  - thus, **global network information is hard to be collected and link failure is difficult to be addressed**
  - while **in a data center network, some programmable architectures have been proposed with a centralized controller**
  - **centralized mechanisms usually employ a global controller**
  - a **controller can collect global information and deal with link failures automatically**, and provide a promising manner for centralized real-time traffic scheduling

© Paulo Ferreira

Comparison of Traffic Scheduling in Wide Area Networks and Data Centers-Differences Between WANs and Data Centers (3/3)

typical centralized architecture is shown in this slide



- **Data center topology is more regular, traffic changes more rapidly, and centralized control is common for data center networks**, which offers more design space for load balancing algorithms

© Paulo Ferreira

Recent Proposals

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers

- Based on the **special traffic characteristics and topologies of data centers**, several traffic scheduling mechanisms have been proposed:
  - we **decompose the design of a data center load balancing mechanism into two main procedures** and **summarize typical methods used in existing literatures for each procedure**

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers- Decomposing a Load Balancing Mechanism (1/5)

- Based on existing mechanisms for data center networks, we conclude that **there are two crucial procedures in a load balancing mechanism**:
  - **collecting congestion information** and **selecting paths**
- Next, we **analyze and categorize the key methods** used in existing mechanisms for the two procedures:
  - **Collecting Congestion Information**
  - **Selecting Paths**
- **Collecting Congestion Information**:
  - a **load balancing scheme requires to determine flow paths** according to congestion information such as link utilization
  - therefore, **a key procedure of a load balancing mechanism is to select one or multiple proper metrics to represent congestion** and transfer the congestion information to the entity that will make load balancing **decisions**
  - by **analyzing existing load balancing schemes**, we conclude the following methods of collecting **congestion information, and classify existing schemes in the table (in the next slide)**

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers- Decomposing a Load Balancing Mechanism (2/5)

- **TCP-based**:
  - **TCP treats packets loss as a signal of network congestion** and will decrease its congestion window as congestion is detected
  - therefore, **some schemes leverage similar signals as TCP or TCP variants to detect congestion**
- **Sending rate**:
  - **the sending rate of flows is a direct value that could be recorded by switches and end-hosts**
  - by **dividing increased sent bytes by the time interval, the average sending rate of each flow could be obtained**
  - **therefore, some schemes measure congestion according to the sent bytes of flows**
  - this method is commonly used in **SDN** network since **OpenFlow switches record sent bytes automatically and the controller could get the value at any time**

Method	Mechanism	Detail
TCP-based	Mahout [14]	Socket buffer
	MPTCP [21]	Modified TCP
	FlowBender [23]	ECN
	CLOVE [26]	ECN
Sending rate	Hedera [13]	Controller polls switches
	MicroTE [15]	Controller polls switches
	Freeway [12]	Controller polls switches
	Fastpass [16]	Controller polls switches
	FDALB [20]	Switch records
	PEFT [22]	Switch records
	CONGA [10]	Switch records
	HULA [25]	Switch records
	DiFS [27]	Switch records
	LocalFlow [30]	Switch records
Switch queue len	Expeditus [88]	Output queue and input queue
	DeTail [28]	Output queue and input queue
	DRILL [34]	Output queue
	CAAR [89]	Output queue

- **Switch queue length**:
  - the utilization of **switch buffer is a direct reflection to the traffic of flows passing that switch**
  - some schemes employ switch queue length in the path to **represent path utilization and detect congestion**

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers- Decomposing a Load Balancing Mechanism (3/5)

- Selecting Paths:
  - **after obtaining congestion information** of data center networks, **load balancing schemes need to schedule arrival flows across different paths to balance load**
  - see table in this slide

Method	Mechanism	Detail
Least congested	Mahout [14]	Least congested in controller
	Freeway [12]	Least congested in controller
	FDALB [20]	Least congested in controller
	PEFT [22]	Least congested in record
	CONGA [10]	Least congested in record
	Expeditus [88]	Least congested in record
	HULA [25]	Least congested in record
	DeTail [28]	Least congested in record
Less congested	MPTCP [21]	Redirect to an under-utilized path
	FlowBender [23]	Redirect to a random path
	DiFS [27]	Redirect to an under-utilized path
	DRILL [34]	A less congested path of two
	Hedera [13]	A path with enough resource
Round-robin	CLOVE [26]	Weighted round-robin
	DRB [29]	Digit-reversed round-robin
	RPS [31]	Original round-robin
	RepFlow [32]	Original round-robin
	TinyFlow [33]	Original round-robin
	Presto [11]	Original round-robin
Integrated method	Fastpass [16]	Determine the path and sending time for each packet
	LocalFlow [30]	Distribute aggregate flows

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers- Decomposing a Load Balancing Mechanism (4/5)

- **Least congested**:
  - implies **recording the congestion paths**
  - the **ideal mechanism for path selection is to assign each flow to the least congested path**
  - therefore, **some schemes collect the utilization information of each path and forward flows to the least congested one**
  - the **performance of this method is highly correlated with the accuracy and responsiveness of the path utilization information**
- **Less congested**:
  - **since it is difficult to collect accurate link utilization in real time**, another method of selecting paths is **moving flows from congested paths to less congested ones**
  - this manner **reduces the overhead to record all path information**

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers- Decomposing a Load Balancing Mechanism (5/5)

- **Round-robin**:
  - the original round-robin manner is to **assign flows one by one to all feasible paths**
  - since it is **agnostic to flow and link information**, it is **easy to make flow collisions in heavy-load scenarios**
  - therefore, some schemes deploy **weighted-round-robin method according to flow length or path utilization to balance traffic**
  - the **performance of this method is correlated with the accuracy of path weight**

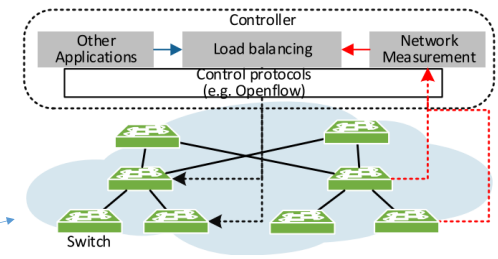
- **Integrated allocation**:
  - in the former three methods, the process of path selection for flows is independent with each other, which may lead to a local optimum
  - therefore, **some schemes aggregate a set of flows and make an integrated allocation to them to achieve a global optimum**

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers- Centralized Mechanisms and Distributed Mechanisms (1/3)

- According to their **methods of scheduling flows, existing data center load balancing mechanisms could be broadly classified into**:
  - **centralized** mechanisms, and
  - **distributed** mechanisms

- Software running on a central controller can **collect congestion information and assign flows to paths based on the global view of the controller**



- For **example**:
  - the controller in the Fig. in this slide **leverages the Openflow protocols to communicate with switches and measures the network through a network measurement application**
  - with the help of measurement application, the load balancing application can easily **obtain global link utilization and traffic information, and uses the information to balance the load**
  - however, **additional communication overhead will be induced to transfer information to the controller**

© Paulo Ferreira

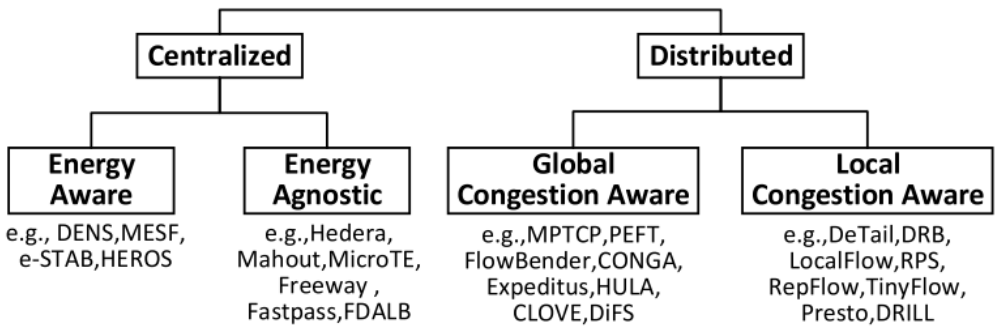
Recent Proposals for Load Balancing in Data Centers- Centralized Mechanisms and Distributed Mechanisms (2/3)

- **Coarse time granularity is another limitation of centralized mechanisms**:
  - most centralized controllers **could not poll information in enough granularity to deal with bursty data center traffic with low cost**

- On the contrary:
  - **distributed schemes select paths locally at hosts or switches**
  - they are **prompt enough to handle traffic bursts in data centers**, but it is **challenging for distributed algorithms to collect global congestion information**
  - furthermore, distributed algorithms are **usually designed for symmetric networks** and they are **difficult to react to topology changes such as link failures and switch crashes**

© Paulo Ferreira

Recent Proposals for Load Balancing in Data Centers- Centralized Mechanisms and Distributed Mechanisms (3/3)



© Paulo Ferreira

Introduction to Cyber-foraging

© Paulo Ferreira

Motivation for Cyber-foraging– how to partition

- **Applications** increasingly **pervade** our daily lives
- Innovation in computing hardware has continually provided users with **more computational resources** in lighter and smaller form factors:
  - laptops have mostly replaced personal computers, and
  - are now in turn being replaced by tablets and smartphones (in some cases)

"How should one best partition applications across mobile platforms and fixed infrastructure?"

- The portability of current mobile devices satisfies this demand by allowing their users to run applications **anywhere at any time**
- However:
  - most mobile applications **do not run in isolation** on the mobile device
  - they **access data and computational resources in the cloud** via wireless (e.g., cellular base stations and WiFi access points)
- Wired infrastructure components such as servers and data stores provide **more computational resources than a mobile device**:
  - they are not limited by stringent size, weight, and battery energy constraints

© Paulo Ferreira

Motivation – thin vs thick

- Current applications typically address this question in an ad-hoc fashion, either by operating in a:
  - **thin-client** mode, or in
  - **thick-client** mode

"How should one best partition applications across mobile platforms and fixed infrastructure?"

- Thin-client mode:
  - all computation/data access is performed at the fixed infrastructure
  - the **mobile device operates only as a terminal**
- Thick-client (or fat) mode:
  - the computation/data are hosted on the mobile device
  - wireless networks are used only to **fetch inputs not available locally**
- A handful of applications adopt a more balanced, but still **static**, **partitioning** of **functionality**

© Paulo Ferreira

Motivation - static vs dynamic

- **Static** partitioning **fails** to account for the **variability** inherent in mobile environments:
  - **wireless network quality** can change by one or more orders of magnitude as users move
  - mobile devices **may become disconnected** from the fixed infrastructure
  - shared computational resources in the cloud **may become overtaxed**
  - nearby computational resources may **become available for opportunistic usage**
  - resources available to different mobile devices can **vary substantially from platform to platform**
  - battery energy and/or cellular bandwidth are **limited resources** and may need to be consumed within a **budget**
  - application **demand** and user **workloads** may **change over time**

"How should one best partition applications across mobile platforms and fixed infrastructure?"

- There is no "**always-best**" **static partitioning** of application functionality
- Thus, any **static partitioning** will inevitably lead to:
  - a **poor user experience** when the actual usage environment varies from that envisioned by the developer

© Paulo Ferreira

Cyber-foraging

© Paulo Ferreira

Cyber-foraging – partition, migration, replication

- It enables the **seamless mobility of data and computation**
- Users are given the **illusion that their applications run entirely locally** on their devices
- This illusion preserves the **always available** access to applications
- Behind this illusion, however:
  - computation and **data may be replicated and migrated** between the several computers
- This infrastructure may either be designated resources:
  - in the **cloud**, or
  - **nearby compute and storage resources** that are **dynamically identified** and **opportunistically exploited**
- As the conditions **changes**:
  - cyber foraging systems **dynamically adjust the partitioning of computation and data**
- As long as sufficient infrastructure **resources can be harnessed** from the environment:
  - cyber foraging can provide users with the **performance** they have come to expect from better-equipped computing platforms

"How should one best partition applications across mobile platforms and fixed infrastructure?"

© Paulo Ferreira

Potential Benefits

- There are many reasons why an application may benefit from:
  - ***executing a computation*** or ***storing data*** on **remote infrastructure** instead of or in addition to the mobile computer on which it is currently executing
  - better **performance**, more **storage**, less **battery** used, app **fidelity**

- There is a fundamental performance **gap between mobile and fixed computers**

Year	Representative server		Representative handheld	
	Processor	Speed	Device	Speed
1997	Pentium II	266 MHz	PalmPilot	16 MHz
2002	Itanium	1 GHz	Blackberry 5810	133 MHz
2007	Core 2	9.6 GHz (4 cores)	Apple iPhone	412 MHz
2012	Xeon E3	14 GHz (2x4 cores)	Samsung Galaxy 3S	3.2 GHz (2 cores)

- Comparing the processing power of representative computer systems in five-year increments including the century crossing (1997–2012):
  - as a **rough estimate**, **processing power is given by the product of core count and clock speed**
  - although both server and mobile computers both show rapid growth in processing power, the **performance gap** between the two remains substantial for every time period examined

© Paulo Ferreira

Examples of Static Partitioning (1/3)

Introduction: https://www.youtube.com/watch?v=Z0gpN7V5zQs

© Paulo Ferreira

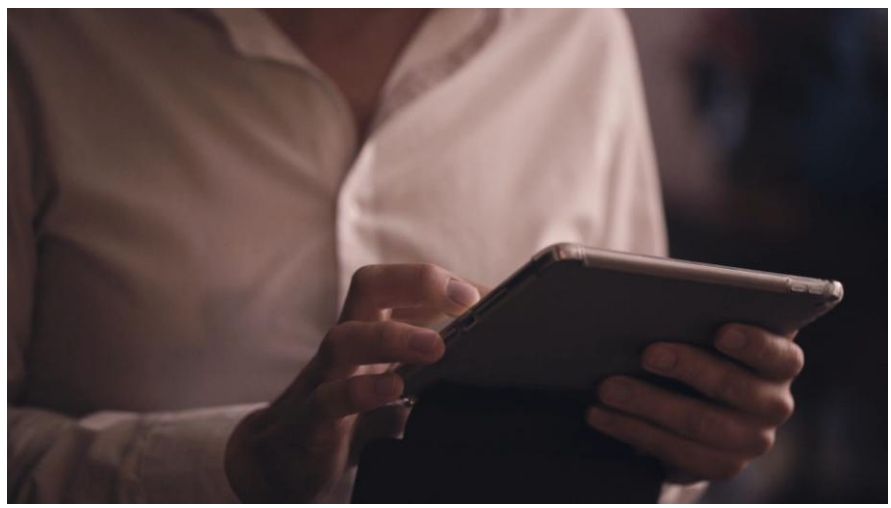
Examples of Static Partitioning (2/3)

Conversation: https://www.youtube.com/watch?v=eP6XqoXrkvk

© Paulo Ferreira

Examples of Static Partitioning (3/3)

Lecture mode: https://www.youtube.com/watch?v=4DfJc--WFvc



© Paulo Ferreira

Benefit – performance

- Due to the **gap** between mobile and infrastructure processing power:
  - **compute-intensive** applications can execute much faster on remote infrastructure
- On the other hand:
  - **interactive activities** that demand few computational resources may execute almost as fast on a mobile computer as they do on a server
- Further, performance is not impacted solely by processor speed:
  - remote infrastructure may offer **more memory and storage**,
  - the ability to **parallelize computation** across multiple cores and servers, or
  - better **network connectivity**

© Paulo Ferreira

Benefit - battery

- Designers strive to make mobile computing systems as **energy-efficient** as possible, for example:
  - employing **hardware power-saving modes**, or by
  - **reducing the scope or quality** of activities performed by mobile applications
- While these measures are essential to **extending battery lifetime** they also:
  - noticeably **degrade** the mobile user experience
  - applications take **longer** to perform interactive activities and produce **lower-quality results**
- Use of fixed infrastructure is an attractive alternative for designers:
  - by **offloading computation or data storage** from a battery-powered computer to a remote computer with wall power,
  - the operational **lifetime of the battery-powered** mobile computer can sometimes be **extended**
  - **without degrading** the user experience

© Paulo Ferreira

Benefit - fidelity

- **Fidelity**:
  - the degree to which the **results produced by an application match the highest quality results** that would be produced given ample computational resources
- Two reasons why a mobile computer operating alone may choose to **reduce application fidelity**:
  - to achieve acceptable **response times** for interactive applications
  - to extend the mobile computer's **battery lifetime**
- For demanding applications:
  - **fidelity reduction** may sometimes be absolutely necessary in order to get the **application to run** on the mobile device at all
  - e.g., a fixed computer may be able to execute an intense computation that a mobile computer is simply incapable of doing locally due to **insufficient memory or storage**
- **Offloading a computation or data query** to a cluster in the cloud provides access to orders of magnitude more resources than a mobile computer has available locally
- There are two choices:
  - **execute** the application **locally** with **reduced fidelity**, or
  - leverage **fixed infrastructure** to provide **full fidelity**

© Paulo Ferreira

Costs of Using Remote Infrastructure (1/3)

- **Offloading application** from the mobile computer **may degrade performance**:
  - the mobile computer must **ship the inputs** to that computation to the remote infrastructure over a wireless network
  - after the computation completes, the mobile computer must **retrieve the results** of the computation
- Unless the computation is asynchronous and not on the critical path of the application, **performance is improved only if the**:
  - time saved by **performing the computation remotely** is greater than
  - the time spent **communicating inputs and outputs** at the start and end of the computation
- It may be quicker to perform the computation on the mobile computer than it would be to perform the computation remotely when:
  - **network latency** between the mobile and remote computers is **high**,
  - **bandwidth is low**, or
  - the amount of **data shipped** per unit of computation is **large**

© Paulo Ferreira

Costs of Using Remote Infrastructure (2/3)

- For **energy savings** the situation is similar:
  - the mobile computer may be able to use aggressive **hardware power-saving modes** while **application activity is performed remotely** (saving energy),
  - it must use a **power-hungry wireless network** to **initiate** the remote activity and **receive** the results
  - it may require more **energy** to **offload activity** to a remote computer than it would require to do the same activity locally

- Offloading application activity to remote infrastructure introduces **security and privacy** concerns:
  - e.g., an **attacker** with physical access to the infrastructure may cause an application to **return incorrect results**, or
  - an **eavesdropper** may monitor the remote computation to **discover private data**

© Paulo Ferreira

Costs of Using Remote Infrastructure (3/3)

Factor	Favorable for execution on the cloud	Favorable for execution on the fog
Network bandwidth	High	Low
Network latency	Low	High
Disparity between compute resources	High	Low
Granularity of computation	Large	Small
Parallelism in computation	High	Low
Memory/storage requirements	High	Low
Size of inputs/outputs	Small	Large
Load on remote computers	Low	High
Wireless network power	Low	High
Sensitivity of activity	N/A	Sensitive

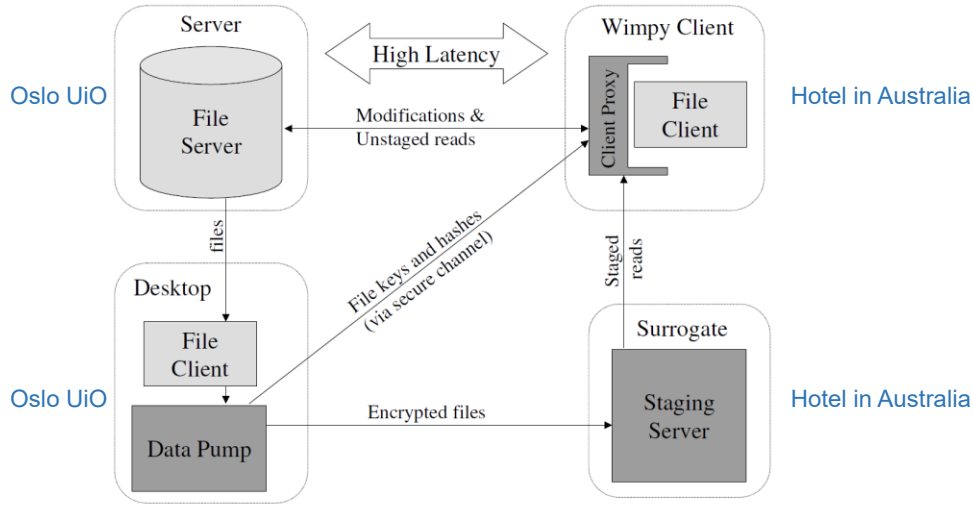
© Paulo Ferreira

Cyber-foraging Complexity

- The **partitioning decision is complex**:
  - a cyber foraging system must **balance numerous goals** (e.g., performance, energy efficiency, fidelity, privacy, and security),
  - it must also evaluate a **large number of factors** (including those shown in the last slide)
- All of the factors listed above may need to be **evaluated dynamically**:
  - **network** conditions and **load** may **change rapidly**
  - factors such as the **granularity of computation** and the **sensitivity of data** may depend upon specific inputs chosen by the user at runtime
- There is simply no **"one-size-fits-all"** choice for all of these factors
- It is **infeasible** to expect that a static partitioning decision can yield optimal or even good results in many environments in which a mobile computer may operate
- Achieving **good results requires** a **dynamic decision**

© Paulo Ferreira

A Data Staging Example (Cyber-foraging)



© Paulo Ferreira

Conclusion

© Paulo Ferreira

Conclusion on Load Balancing

- We addressed **load balancing**, which is becoming an **important research area that enables reducing completion time of flows, maximizing bandwidth utilization, and saving energy** for data center networks
- First, we presented **data center network topologies, traffic characteristics, and main properties and objectives** for load balancing mechanisms
- Then, the **differences between load balancing in data centers and traffic engineering in WAN** were presented
- Next, **recent proposals of load balancing were surveyed**:
  - we broadly classified the proposals into those that use a **centralized controller** to balance load and those that run on **distributed** network devices
- As **large energy consumption is a big issue** in data centers, we also summarized energy-aware load balancing schemes in data center networks

© Paulo Ferreira

Conclusions on Cyber-foraging

- It's a **hot topic** and a research concept which has already practical use cases (**mostly with static partitioning**)
- Cyber-foraging stands at the confluence **of two strong trends** in modern community:
  - **every day more people access data and run computation from their mobile computers**, and
  - **data and computation is increasingly distributed among those devices and the cloud**
- The **large variation in mobile/fog computing environments** argues that functionality should by **dynamically**, rather than statically, partitioned among mobile and cloud computers
- The **limitations of locating computation in both cloud data centers (e.g., communication latency) and on mobile/fog devices (e.g., limited resources):**
  - create the need to insert a surrogate computational tier between these extremes that can mitigate both sets of limitations

© Paulo Ferreira