Unicorn: Unified Resource Orchestration for Multi-Domain, Geo-Distributed Data Analytics

Qiao Xiang[12], Jace Liu[1], Harvey Newman[3],
Tony Wang[12], Y. Richard Yang[12], Jensen Zhang[1]
[1] Tongji University, [2] Yale University,
[3] California Institute of Technology,

November, 2017, INDIS Workshop, Denver, CO

Background

- Data-intensive applications rely on clusters of heterogeneous servers as the major computing platform.

- **Missing:** a unified framework to manage a large set of distributively-owned, heterogeneous resources for multi-domain data analytics.

- **Members**: worldwide multi-organizational collaboration among Caltech, Tongji University, Tsinghua University, Yale University, the OpenDaylight ALTO team and the Kytos team.

Example Design Setting: Large Hadron Collider (LHC)

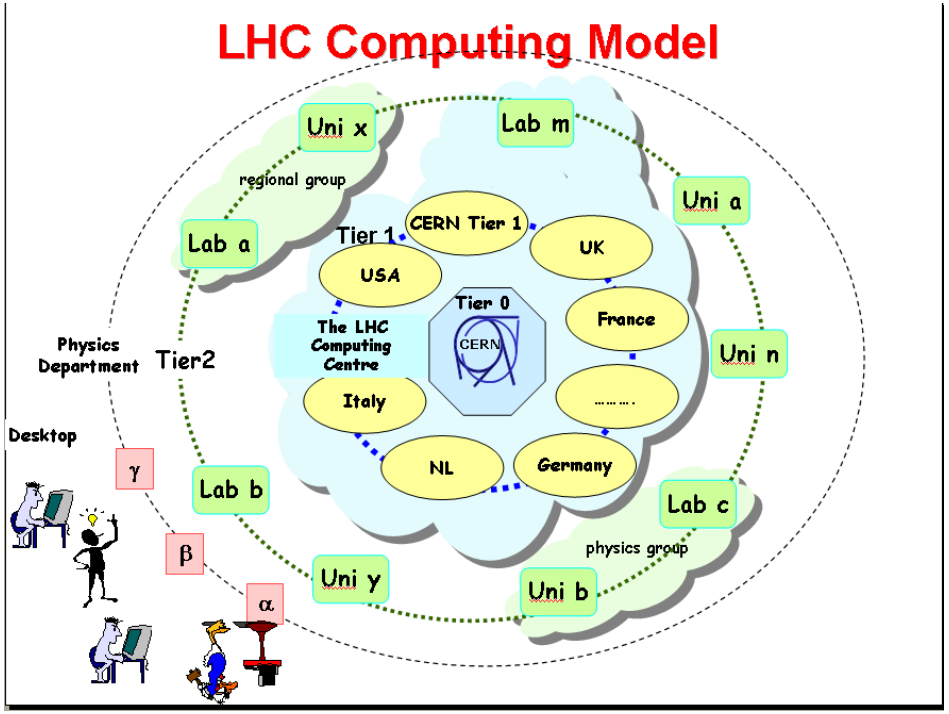
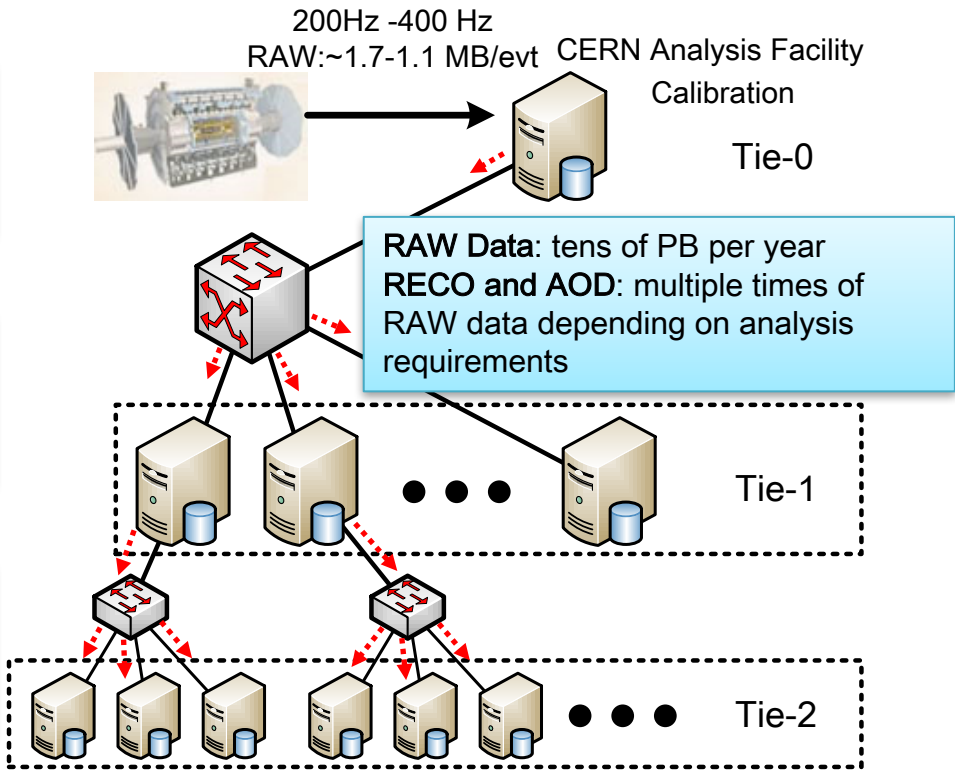


Figure source: cern.ch

The Compact Muon Solenoid (CMS) Computing Model

- Large raw datasets from LHC at the Tier-0 site
- RECO and AOD datasets are distributed to Tier-1 sites
- RECO, AOD and simulation datasets are transferred among Tier-1~3 sites for analysis.



Resource Orchestration in CMS: Challenges

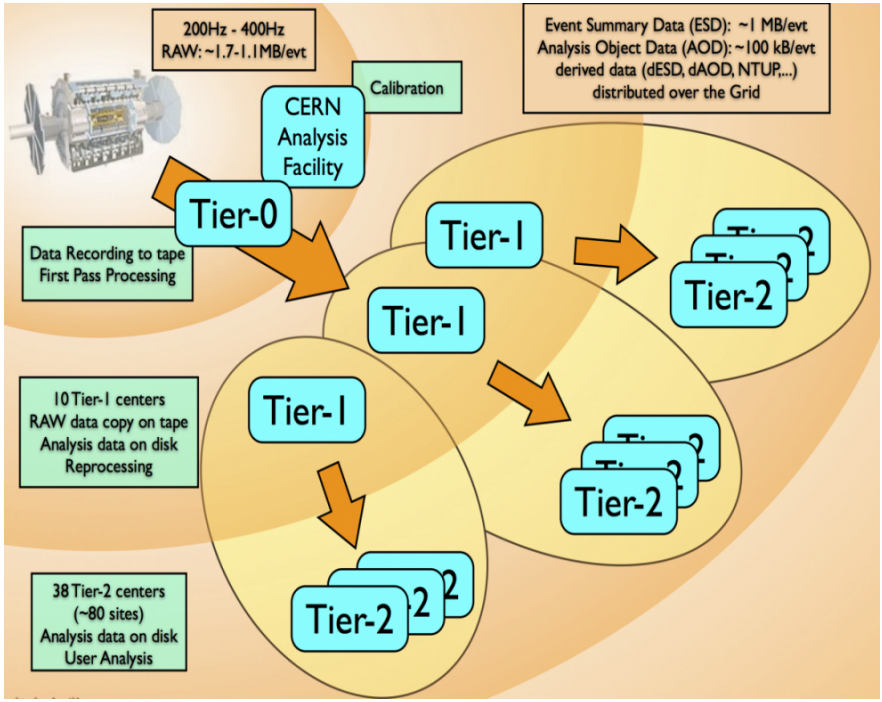


Figure source: cern.ch

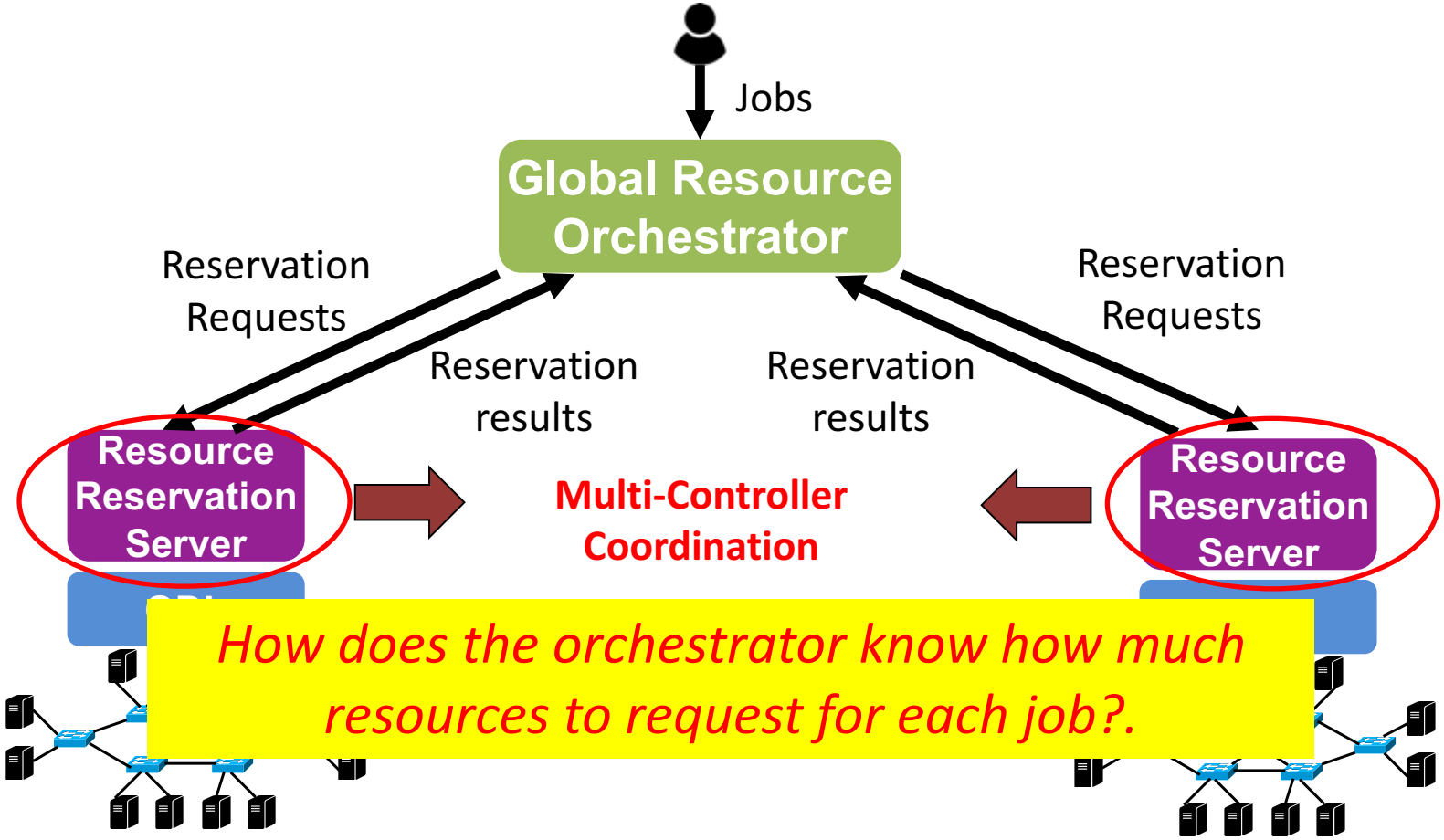
- It is a multi-domain science network.
- Different domains (resource providers) provide heterogeneous resources.
- Different resource providers use different controllers to manage the resources, especially the networking resource, e.g., OpenDaylight and Kytos.
- Different jobs in the network.
  - PB dataset transfers
  - Various HEP analytics

Multi-Domain Resource Orchestration: Design Requirements

- **Multi-controller coordination**.
  - Resource providers, which use different network controllers (e.g, OpenDaylight, Kytos, ONOS, and etc.), can communicate and coordinate the orchestration process through a unified interface.
- **Consistent operation paradigm**.
  - Effic… g.
  - Fast…
- **Autonomy and privacy of resource providers**.
  - Resource providers can make and practice their own resource supply strategies with control of privacy.

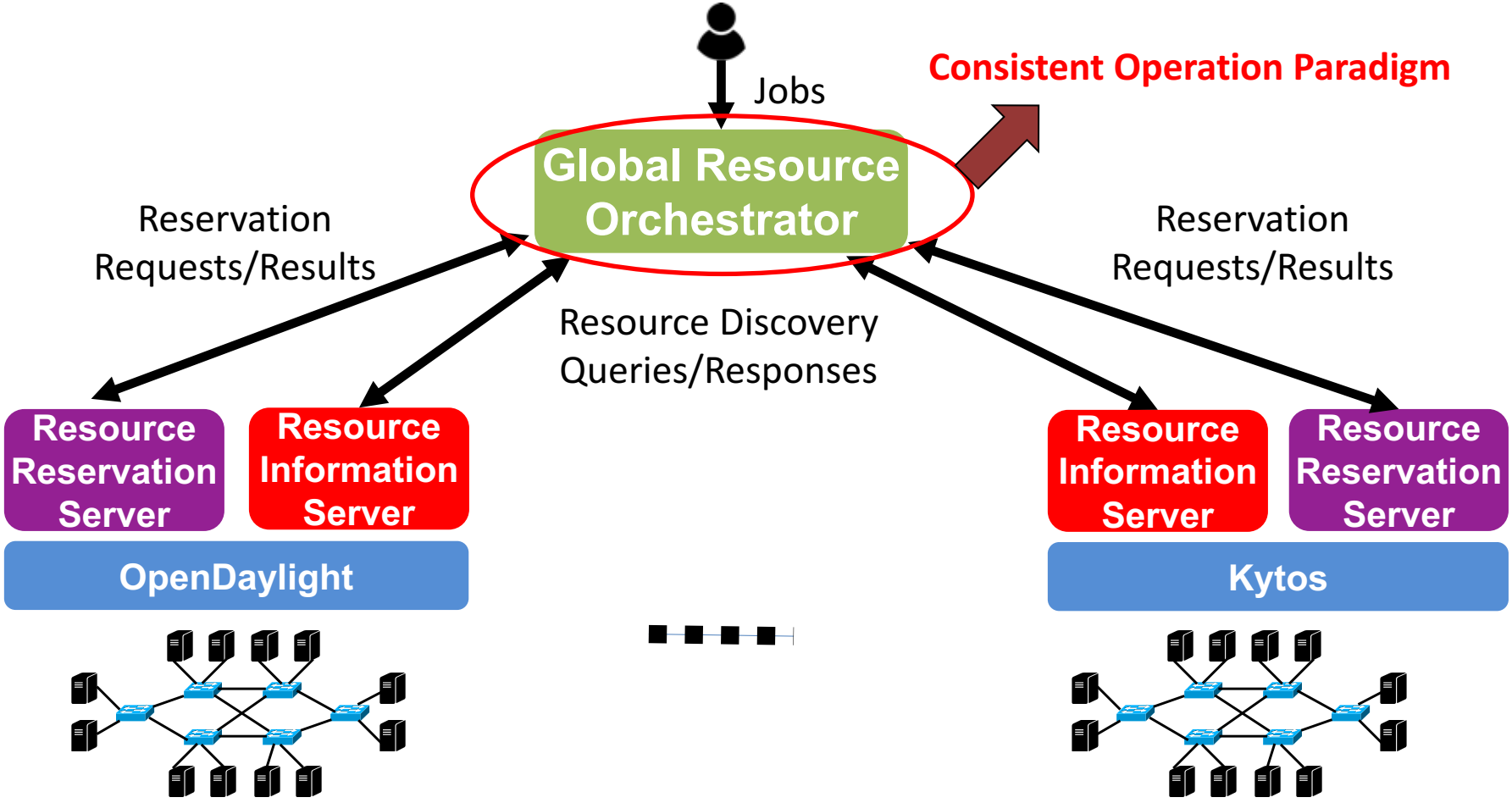
***Unicorn**: a multi-domain, multi-controller (MDMC) resource orchestration system*

Unicorn: An MDMC Resource Orchestration System



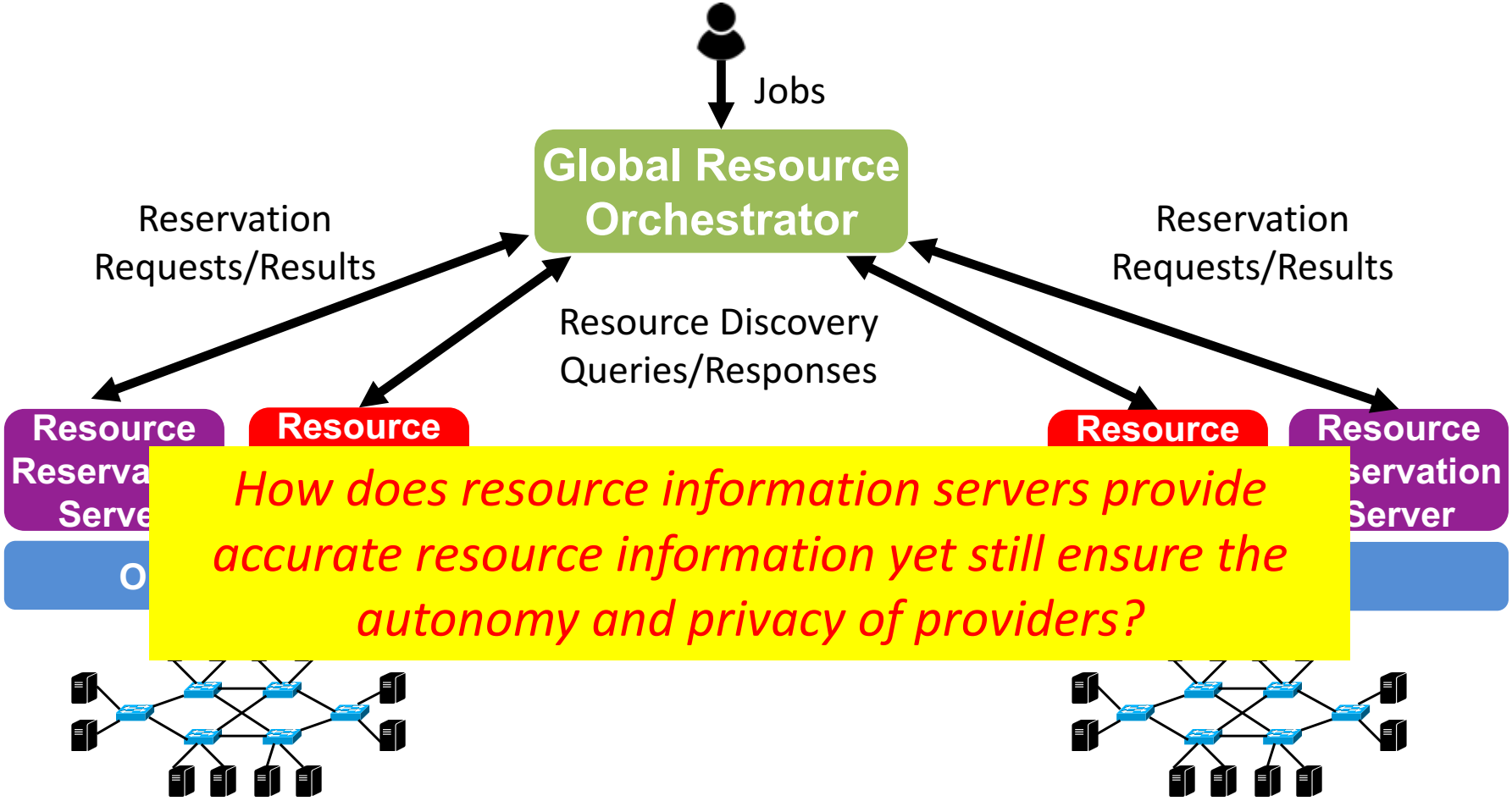
- Users send jobs to a logically centralized orchestrator;
- The orchestrator sends resource reservation requests to different domains;
- The reservation servers, running on top of different controllers, process the requests and return the result (success/fail).

Unicorn: An MDMC Resource Orchestration System



- Add resource information servers to provide such information of each domain.
- The orchestrator uses the queried resource information to compute the optimal resource reservation requests, and send to the reservation servers.

Unicorn: An MDMC Resource Orchestration System



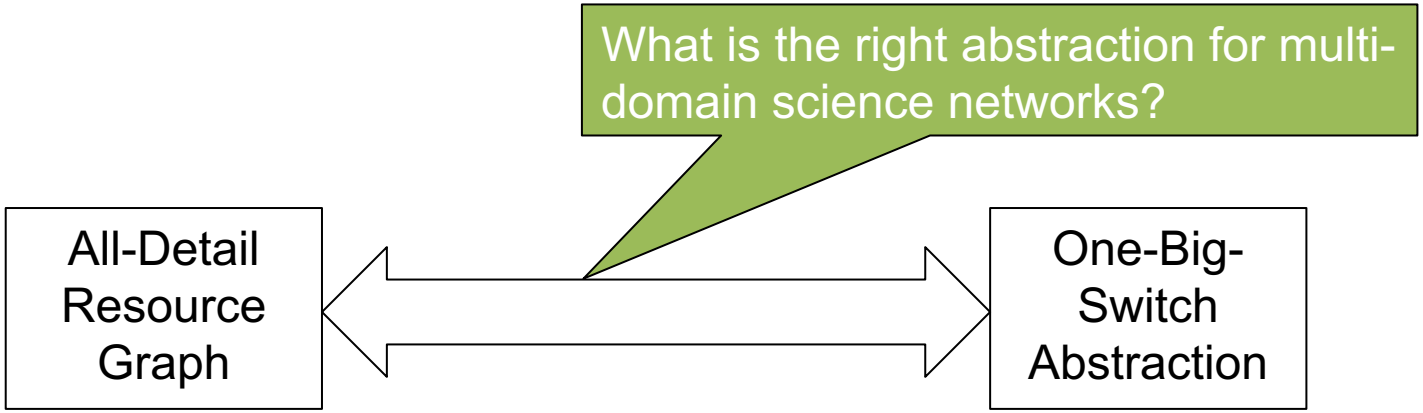
- Consistent operation paradigm
- Multi-controller coordination
- Provider autonomy and privacy

- Global orchestrator
- Servers with unified interfaces
- ?

Resource Information Server: Related Work

- All-detail resource graph
  - Examples: HTCondor, Mesos, YARN, etc.
  - Nodes: computing/storage resources
  - Links: networking resources
  - Limitation
    - Reveal all details of resources, compromising the privacy of clusters in the multi-domain setting.
    - Resource supply heterogeneousity and dynamicity lead to high overhead.
- Alternative design: one-big-switch abstraction
  - Example: P4P/ALTO.
  - Limitation: cannot reveal the shared bottleneck resources between analytics tasks, leading to resource overloading and slow convergence.

Resource Information Server: Solution

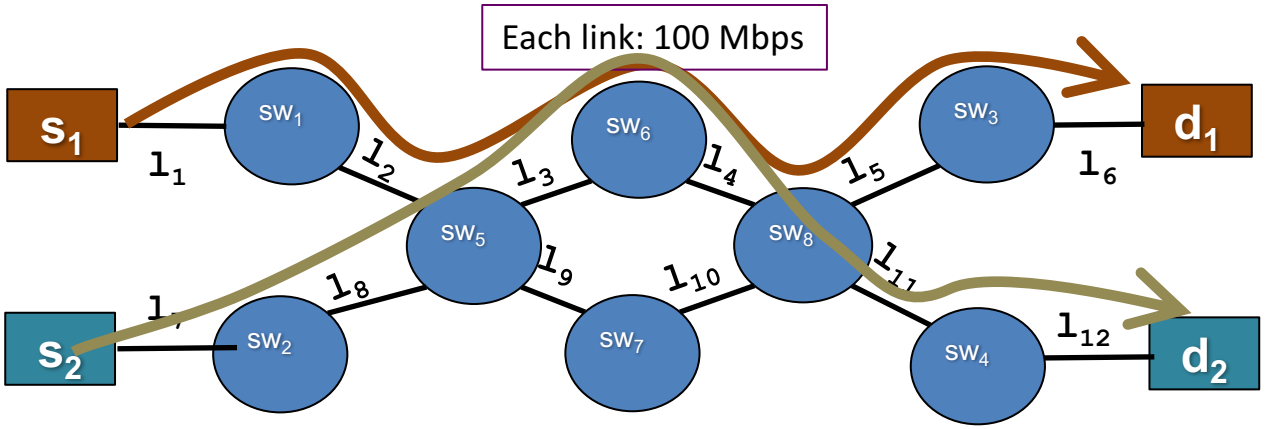


Extremely detailed;
Compromised privacy;
High overhead.

Extremely abstract;
Cannot reveal shared bottleneck resources.

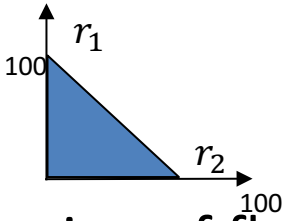
- **Basic idea**: instead of the more limited graph model to represent resource availability, mathematical programming, such as linear programming, is a more general, abstract constraint representation.
- We refer to this feasible region representation as resource state abstraction (ReSA).

Resource State Abstraction (ReSA): Example



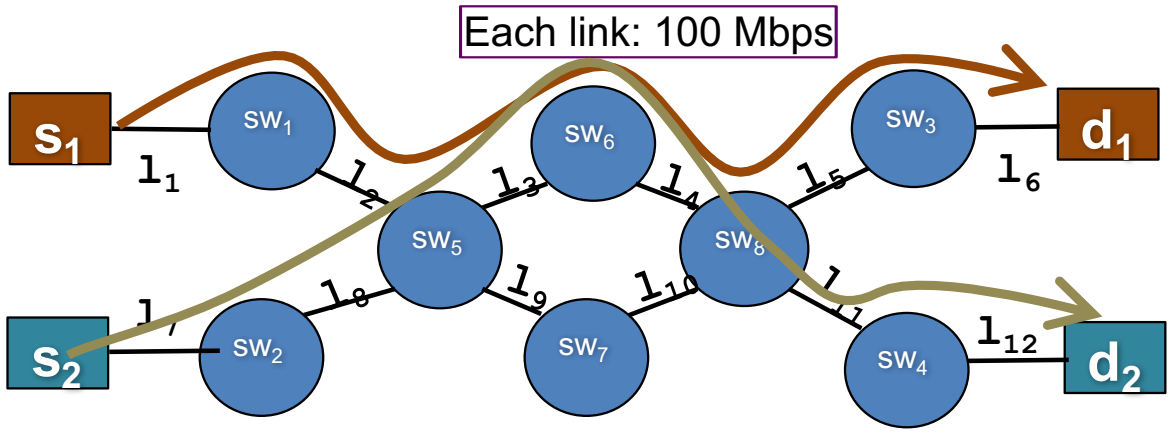
- For each link, use a linear constraint to represent the bandwidth sharing among flows that use this link.

$$r_1 \le b_i, \forall\, i \in \{1,2,5,6\}$$
$$r_2 \le b_i, \forall\, i \in \{7,8,11,12\}$$
$$r_1 + r_2 \le b_i, \forall\, i \in \{3,4\}$$

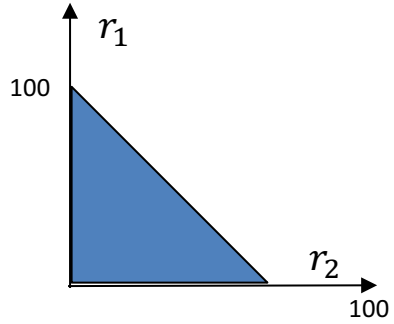


- Geometrically, ReSA is the feasible region of flow rates defined by these linear constraints.
- However, some constraints are redundant, i.e., the feasible region of flow rates will not change without these constraints.

Minimal, Equivalent ReSA: Example



$$r_1 \le b_i, \forall\, i \in \{1,2,5,6\}$$
$$r_2 \le b_i, \forall\, i \in \{7,8,11,12\}$$
$$r_1 + r_2 \le b_i, \forall\, i \in \{3,4\}$$



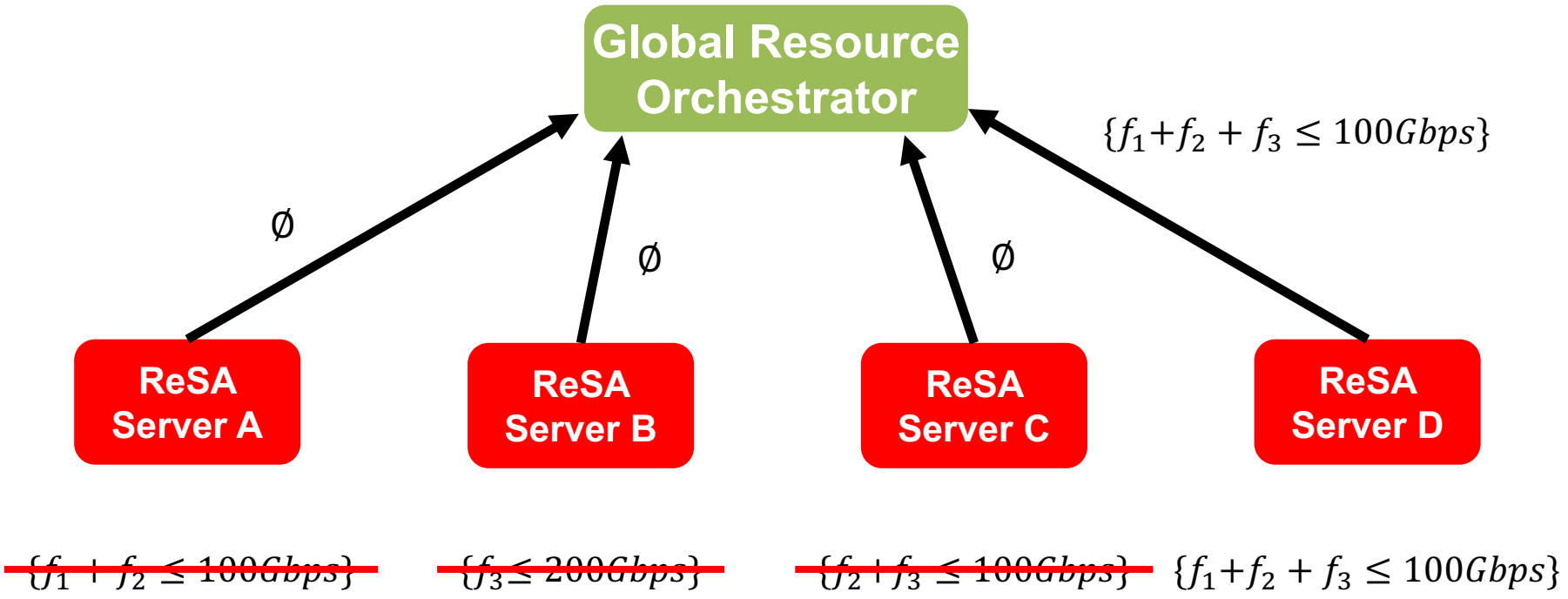
$$r_1 + r_2 \le 100\ Mbps$$

- Minimal, equivalent ReSA reveals shared bottleneck resources.

ReSA for Multi-Domain, Resource Discovery

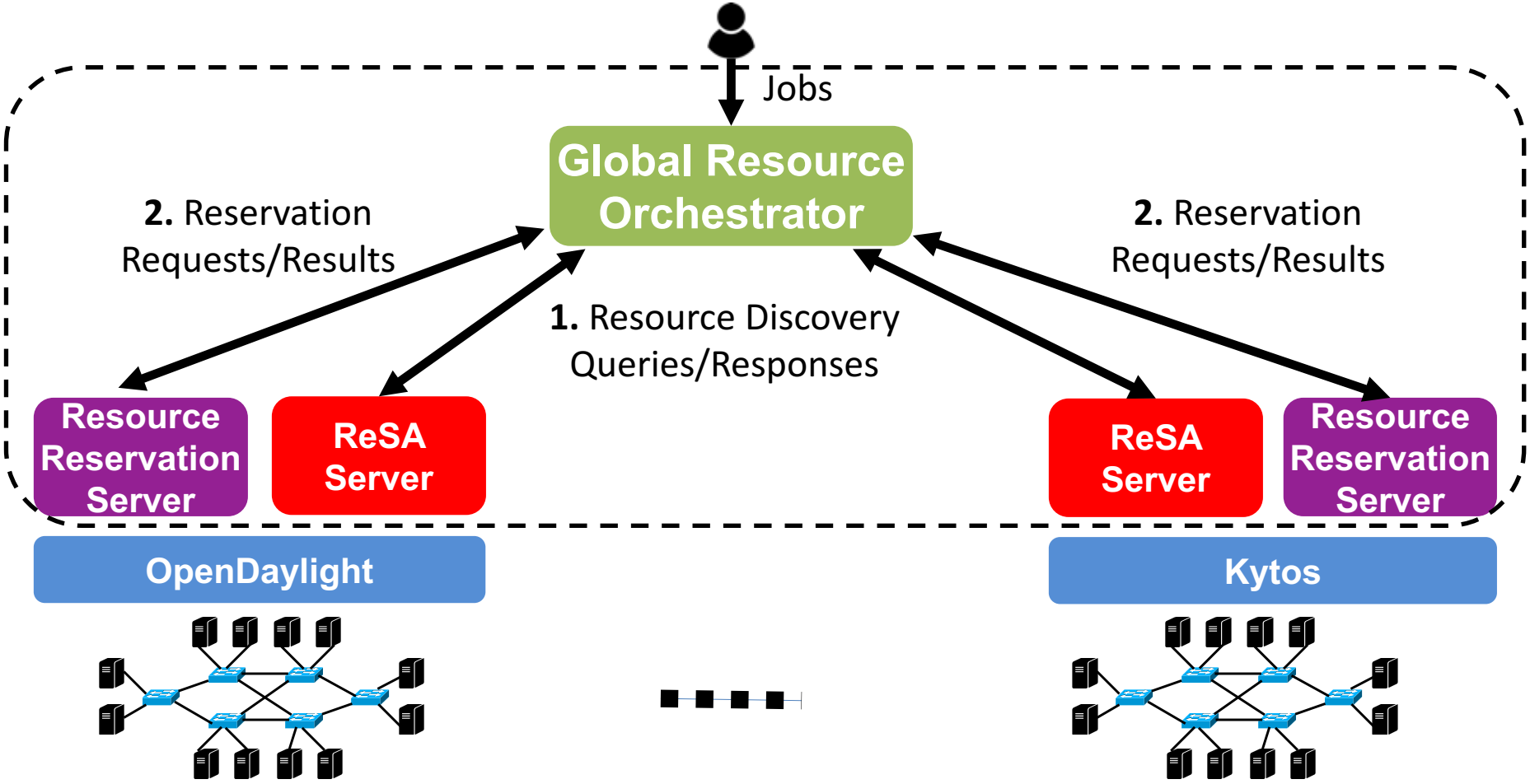
- Accurate, efficient discovery process.
  - Two-phase discovery decomposition.
  - **Path query**: find all the domains it passes through for each job.
  - **Resource query**: ReSA query for all jobs entering the same domain.
- Minimal information exposure of multiple resource providers.
  - Secure multi-party computation.
- Dynamic update of resource availability.
  - Server-side event.

Minimal Information Exposure of Resource Providers



- **Basic idea**. a secure multi-party computational geometry protocol to decide the redundancy of each linear inequality using vertex enumeration and halfspace test.
- ReSA servers from different domains do not reveal their own set of linear inequalities to others during the protocol.

Putting Pieces Together



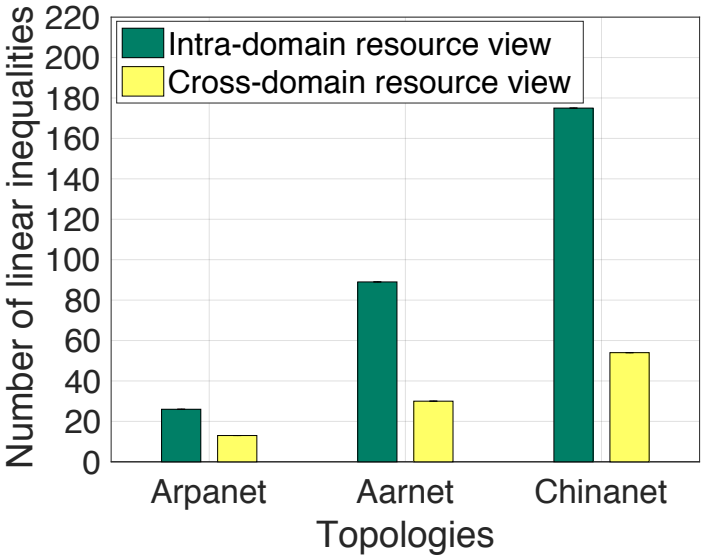
- Multi-domain orchestration
- Multi-controller coordination
- Provider autonomy and privacy

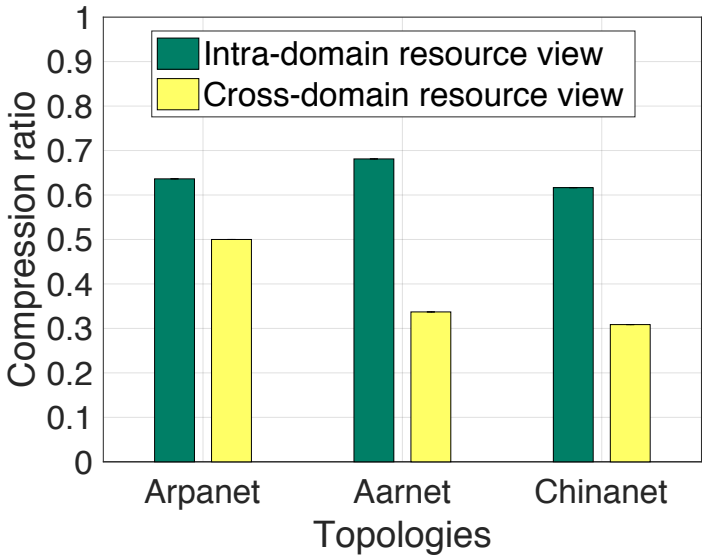
- Global orchestrator
- Servers with unified interfaces
- Resource state abstraction

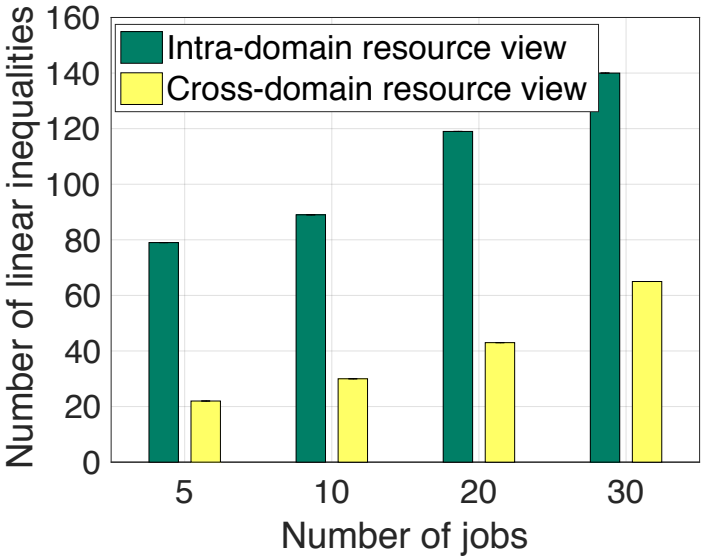
Unicorn Implementation

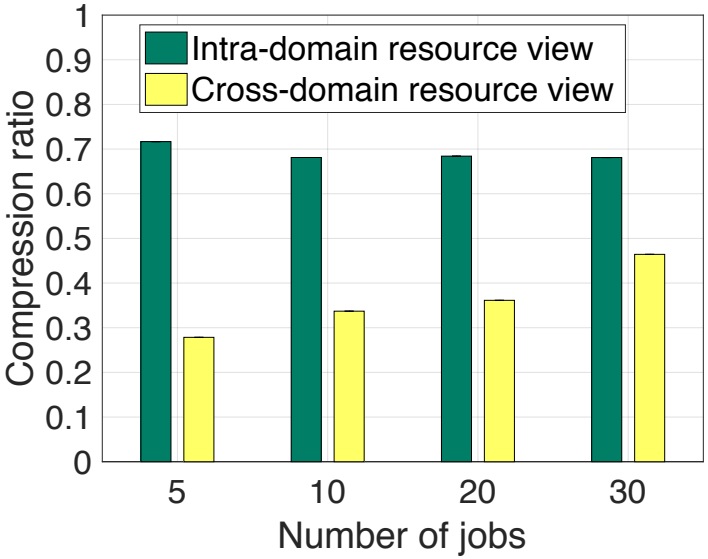
- Orchestrator: ~2700 LoC Python code
- ReSA server: ~2500 LoC Java code
- Resource reservation server:
  - fast data transfer (FDT), FireQoS, OpenvSwitch, etc.
- Network controllers: OpenDaylight, Kytos
  - ONOS and Ryu are under development

Evaluation









Please refer to our paper for more details.

Summary

- **Unicorn**: a multi-domain, multi-controller resource orchestration system
  - Global orchestrator to achieve **consistent operation paradigm**.
  - Servers with unified interfaces enable **multi-controller** coordination.
  - **Resource state abstraction**: a set of linear inequalities to represent resource availability, achieving accurate, minimal information exposure of res

Demonstration at SC17

When: 2-3pm on Tuesday and Wednesday

Where: booth 663 (Caltech Booth)

What: multi-domain resource orchestration across multiple booths and wide area networks.

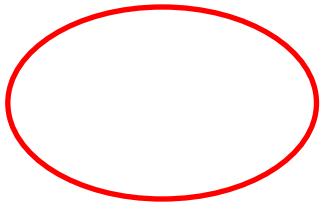
Contact: Qiao Xiang (qiao.xiang@yale.edu)

Backup Slides

Resource State Abstraction (ReSA)

- **Basic idea**: instead of the more limited graph model to represent resource availability, mathematical programming, such as linear programming, is a more general, abstract constraint representation.

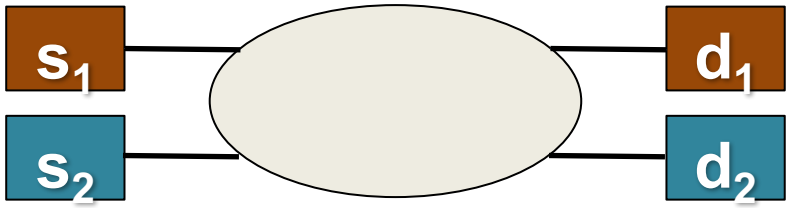
Objective of App k subject to ,



- Introduce mathematical programming constraints, representing feasible regions, as a more powerful, flexible representation of resource availability.
- We refer to this feasible region representation as resource state abstraction representation.

One-Big-Switch Abstraction: Example

- Two flows
  - f_1: $\{s_1, d_1\}$
  - f_2: $\{s_2, d_2\}$

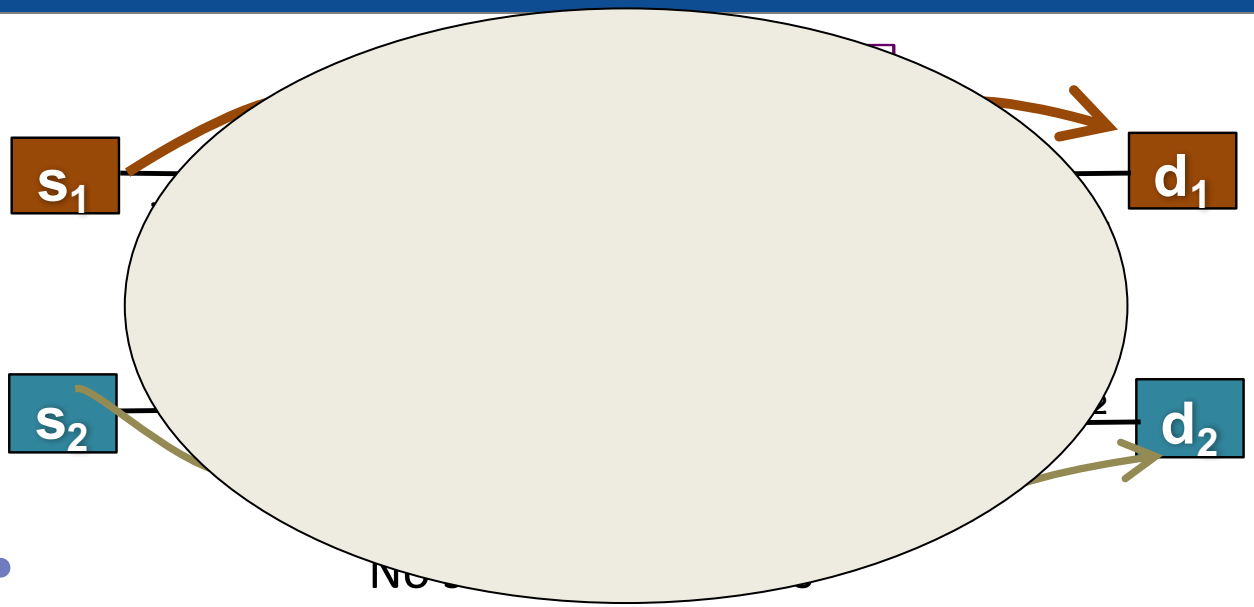


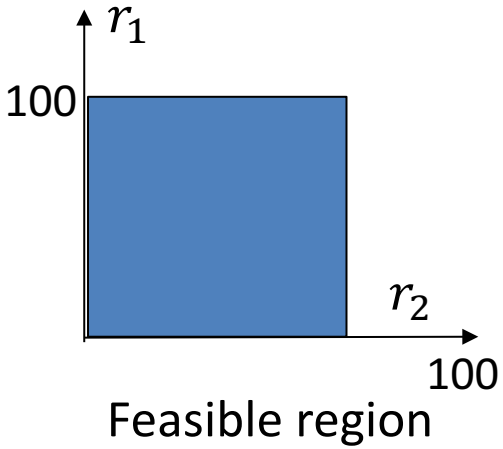
- Resource information provided by one-big-switch abstraction

$$r_1 = 100Mb/s$$
$$r_2 = 100Mb/s$$

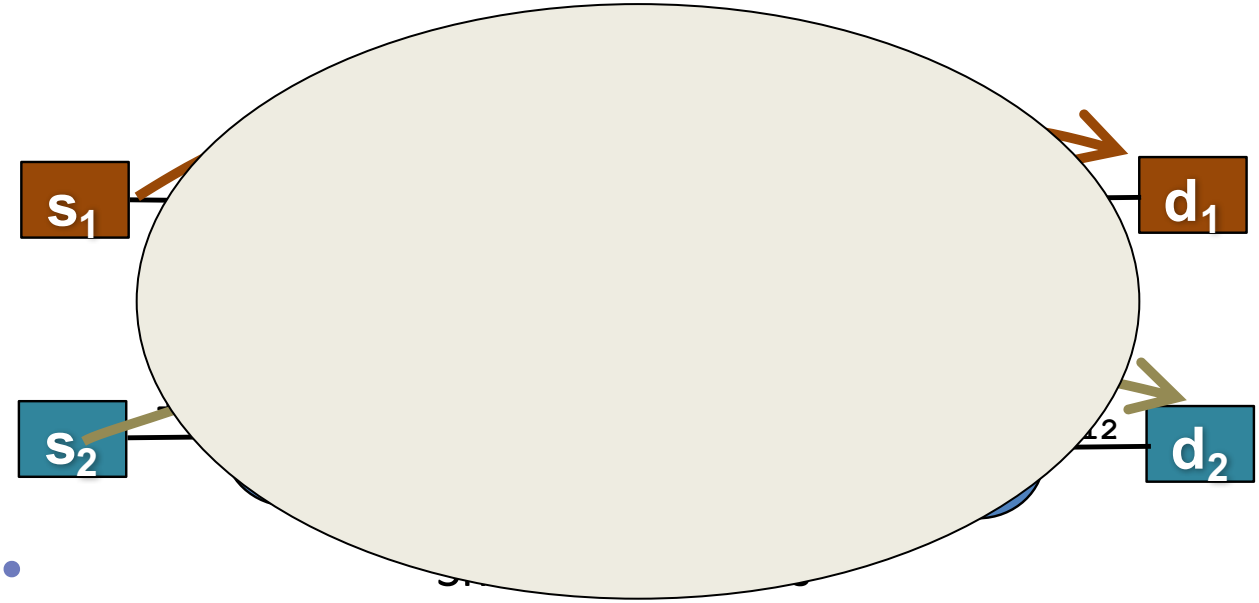
- The view provided by one-big-switch abstraction can have ambiguity.

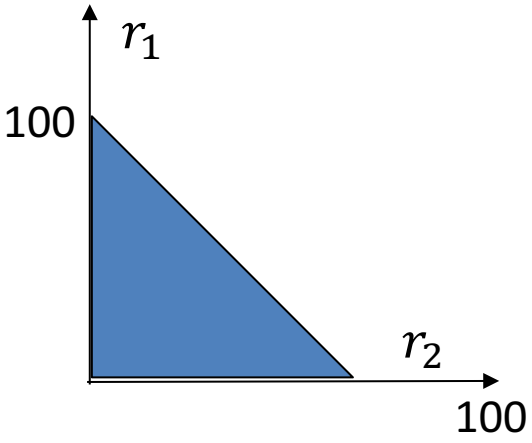
One-Big-Switch Abstraction: Ambiguity





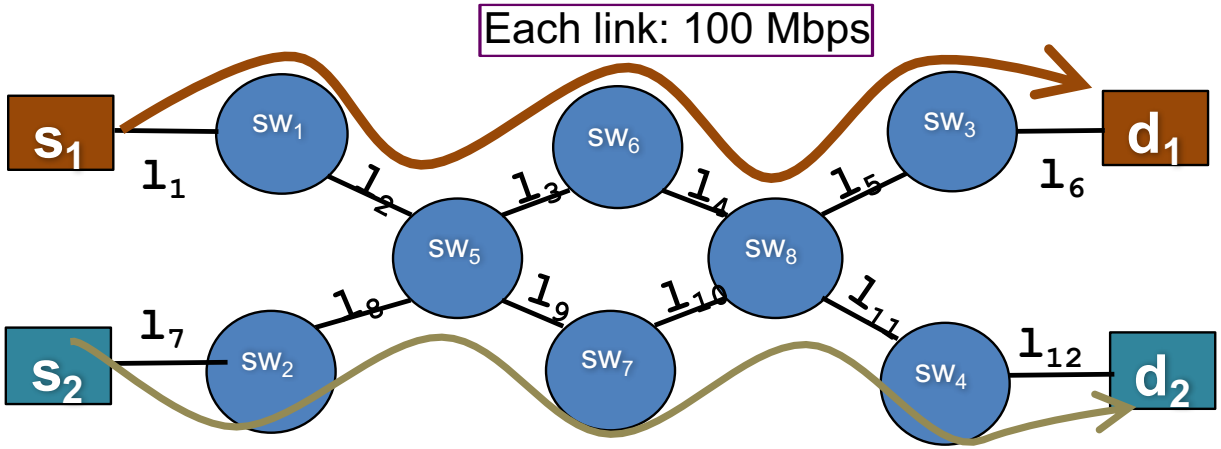
Feasible region





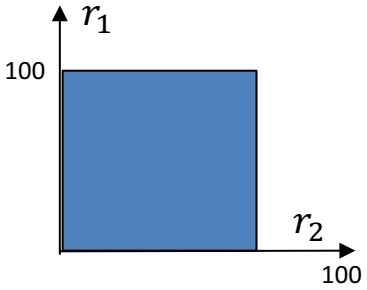
Feasible region

Minimal, Equivalent ReSA Example - 1



$r_1 \leq c_i, \forall\, i \in \{1,2,3,4,5,6\}$
$r_2 \leq c_i, \forall\, i \in \{7,8,9,10,11,12\}$

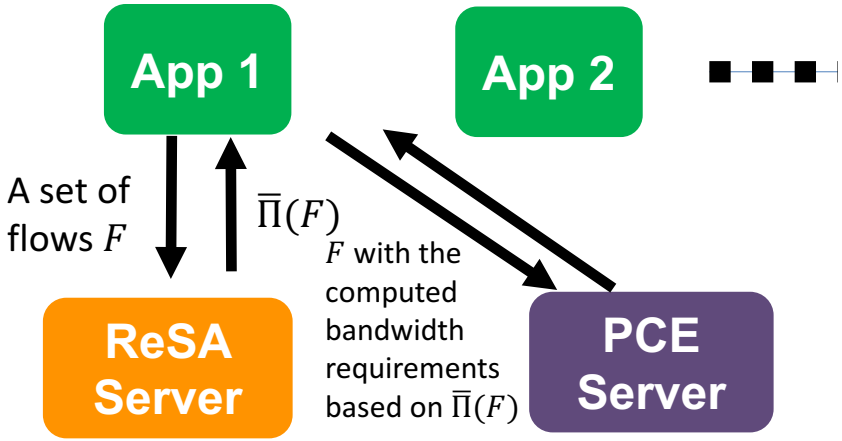
$r_1 \leq 100\ Mbps$
$r_2 \leq 100\ Mbps$

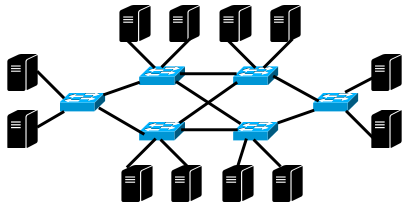


- When there is no shared resource among flows, minimal, equivalent ReSA reduces to the one-big-switch abstraction.

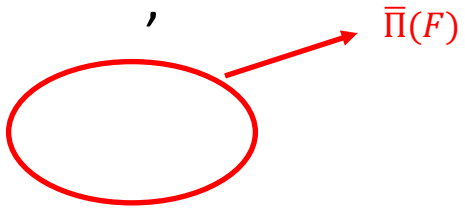
How ReSA Works [2][3]

1. Applications (clients) send a request for a set of flows F.
2. The RSA server collects network information; calculates the minimal, equivalent ReSA $\overline{\Pi}(F)$ for this request; and returns to the client.
3. Applications use $\overline{\Pi}(F)$ as constraints to compute the bandwidth requirement for flow set F, and send the request to the PCE server.





subject to ,



[2] Gao, *et.al.*, "ORSAP: Abstracting routing state on demand", poster, in IEEE ICNP 2016.
[3] Gao, *et.al.*, "NOVA: towards on-demand equivalent network view abstraction for network optimization", in IEEE/ACM IWQoS 2017.

Accurate, Efficient Resource Discovery

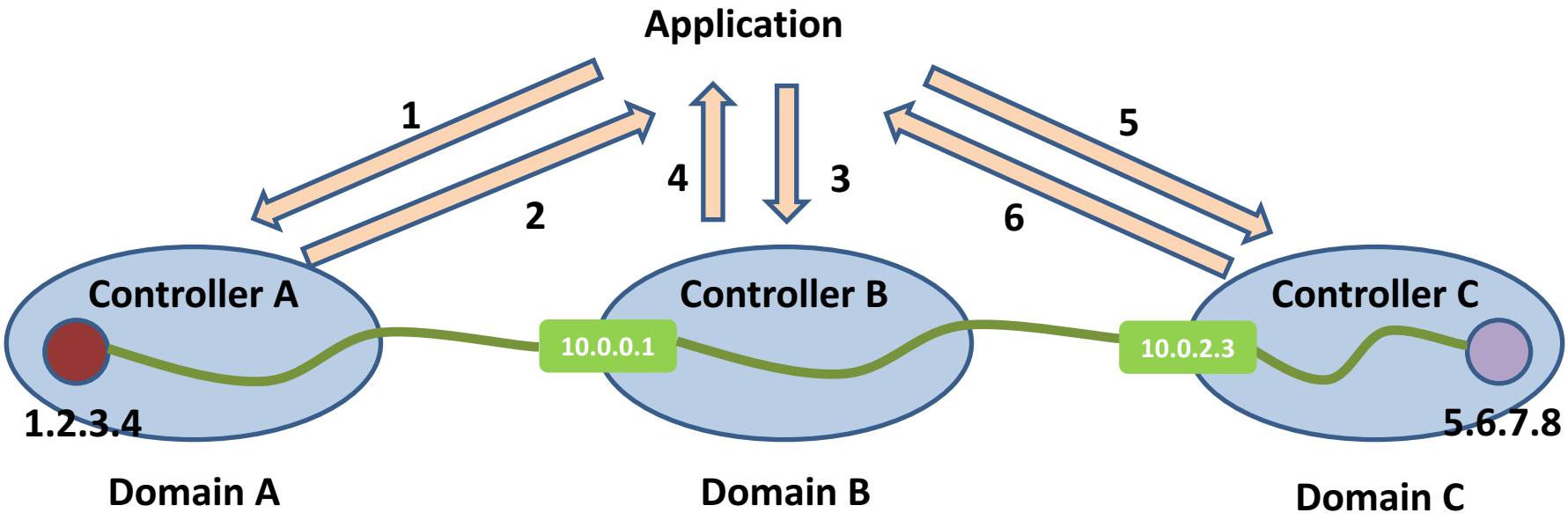
- **Basic idea**. Two-phase discovery decomposition: path query and resource query
- **Path query**: for each job, find all the domains it passes through (i.e., domain-path).
  - The outcome is equivalent to the set of all jobs enter each domain.
  - **Lemma**: path query requires **no additional information exposure** from each domain than current inter-domain routing protocols, i.e., BGP.
- **Resource query**: for each domain, find its accurate resource availability for all the flows that enter this domain (i.e., ReSA).

Path Query

- Existing multi-domain routing protocols, e.g., BGP, provide the information to construct the domain path.

- **Lemma**: path query requires no additional information exposure from each domain than current inter-domain routing protocols, i.e., BGP.
  - Proof: for any flow, the ingress point of each site it passes must be known by the last-hop site to forward the flow.

Path Query Example



1. pQuery([{1.2.3.4, 5.6.7.8}], null) //null means site A is where the source resides in.
2. pResponse: [10.0.0.1]
3. pQuery([{1.2.3.4, 5.6.7.8}], 10.0.0.1)
4. pResponse: [10.0.2.3]
5. pQuery([{1.2.3.4, 5.6.7.8}], 10.0.2.3)
6. pResponse: [null] // reaches the destination site.

Schedulability of ReSA View

- **Proposition**: when the view represented by ReSA satisfies one of the following conditions:
  - resources represented in the original set of constraints C can be fully controlled reserved on the edge, i.e. , all the attributes of each resource can be reserved at end hosts;
  - all the attributes computed in C', the minimal, equivalent ReSA, can be fully fully controlled on the edge;

the RSA view provides a full schedulability of resources to a logically centralized resource orchestrator.

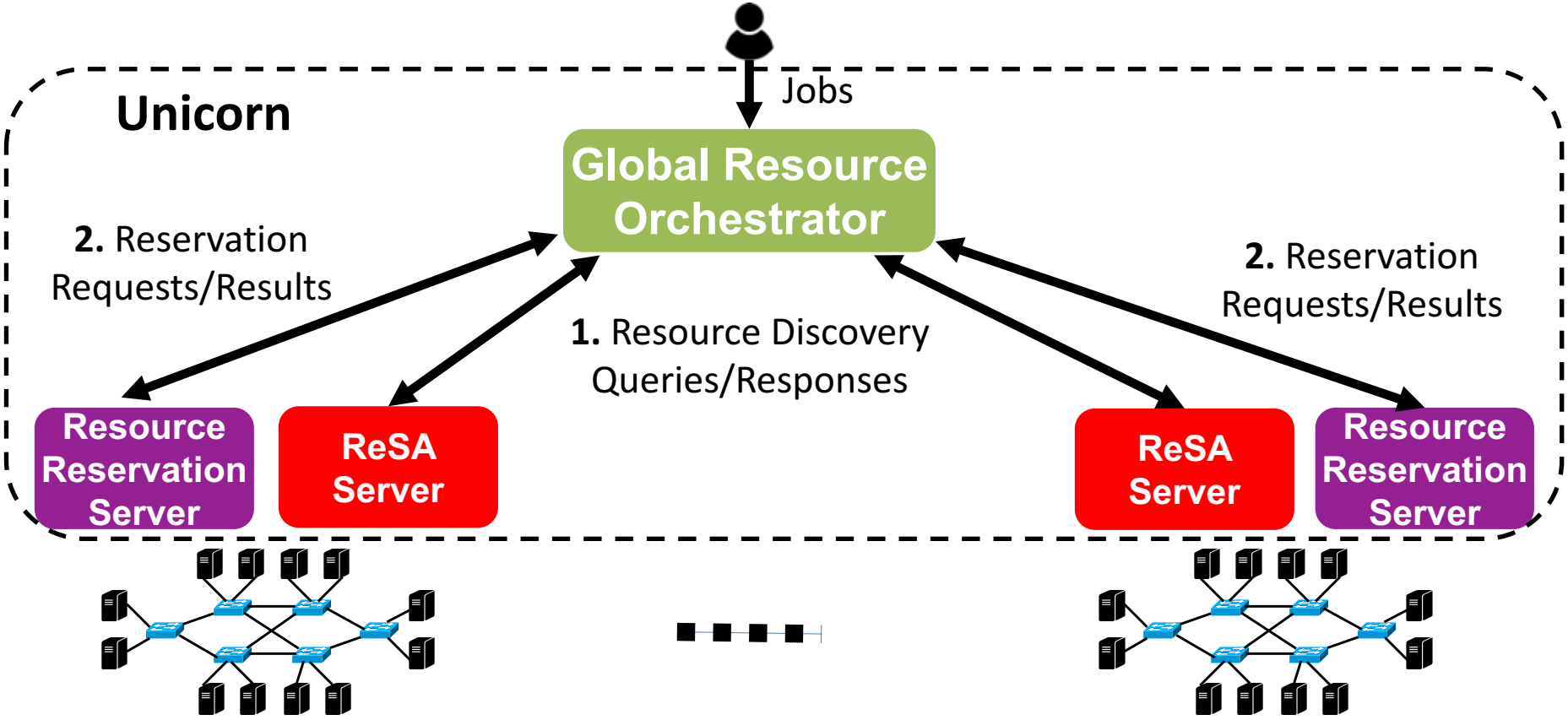
***Question**: what if resources are not controlled at end hosts, e.g., networking resources are controlled by TCP congestion control mechanisms?*

Minimal, Equivalent ReSA

- **Equivalence**: two set of linear constraints Π and $\overline{\Pi}$ are equivalent if their corresponding feasible regions are identical.

- **Minimal, Equivalent ReSA Problem**. given the raw resource state represented by a set of linear constraints Π: $\{\boldsymbol{Ax} \leq \boldsymbol{b}\}$, find $\overline{\Pi}$, the minimal subset of Π that is equivalent to Π.
- **MECS Algorithm**:
  - Iteratively select c: $a^T\ \boldsymbol{x} \leq b \in \Pi$ and solve:
    $$y = \max a^T \boldsymbol{x}, s.t., \Pi \backslash \{c\}$$
  - If $b < y$, put c into $\overline{\Pi}$.
  - A polynomial-time algorithm with proved optimality.

Unicorn Architecture



Resource Information Server: Related Work

- All-detail resource graph
  - Examples: HTCondor, Mesos, YARN, etc.
  - Nodes: computing/storage resources
  - Links: networking resources

- Limitation
  - Reveal all details of resources to applications, compromising the privacy of resource providers.
  - Resource supply heterogeneousity and dynamicity lead to high overhead.

Resource Information Server: Related Work

- Alternative design: One-Big-Switch abstraction
  - Example: P4P/ALTO [1].
  - Combine the objective of applications and providers.
  - Decouple the decision variables from applications and providers in the constraints using prime-dual decomposition.
  - The interface between applications and providers is the dual variables.
- Limitation: cannot reveal the shared bottleneck resources between analytics tasks.
  - An iterative approach which takes time to converge.
  - Cannot prevent applications from overloading the resources.

[1] Xie, *et.al.*, "P4P: Provider portal for applications.", in SIGCOMM 2008.