CS 644: Introduction to Big Data

Daqing Yun (daqing.yun@njit.edu)
New Jersey Institute of Technology

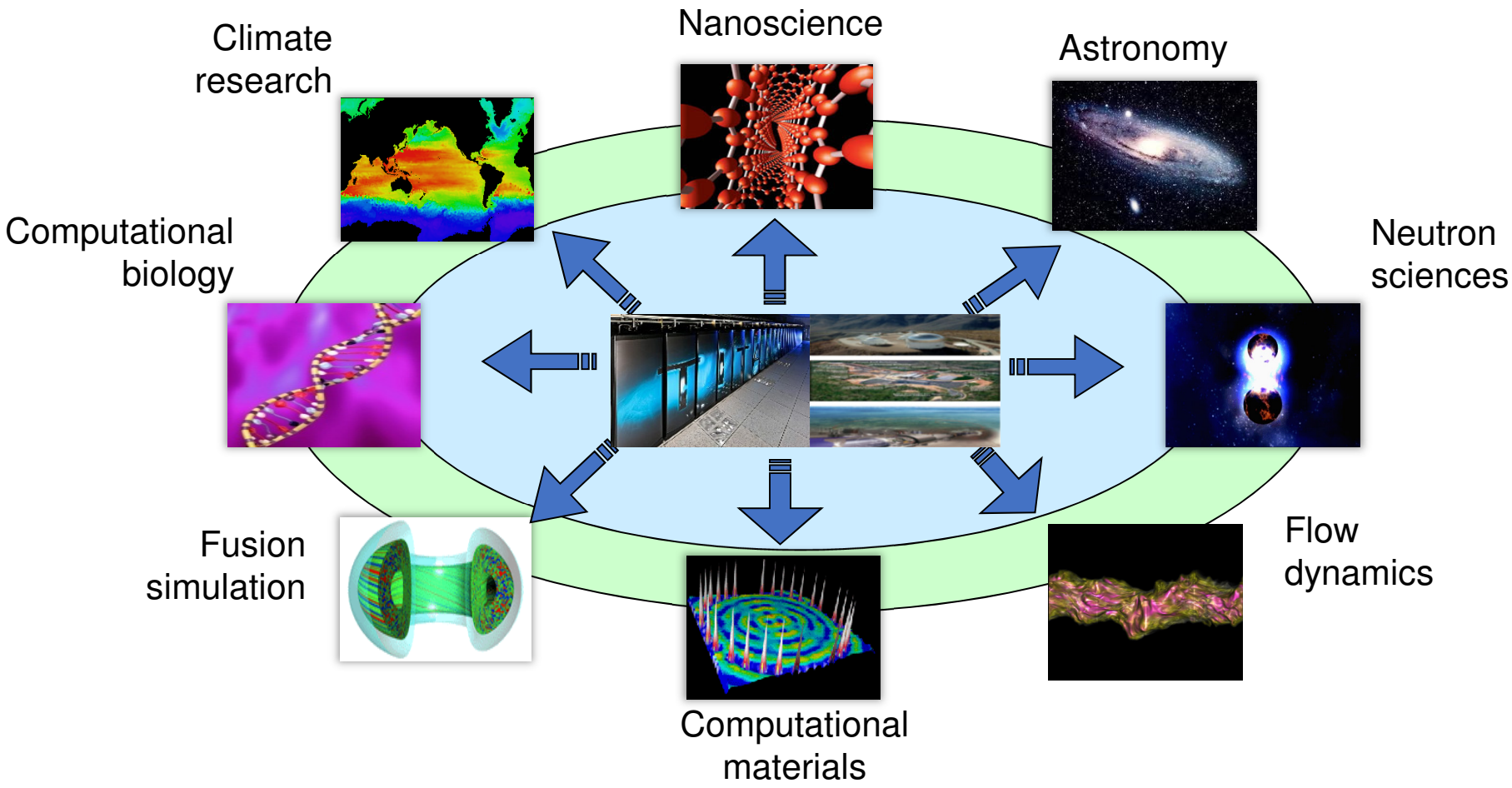
Mira SC@ANL

Outline

- Introduction
- Challenges & Objectives
- A Three-layer Architecture Solution
  - Enabling Technologies: networking and computing
- Computing for Big Data
  - Workflow Management and Optimization
    - Workflow Mapping
    - On-node Scheduling
- Networking for Big Data
  - Software-defined Networking (SDN)
  - High-performance Networking (HPN)
    - Bandwidth Scheduling
    - Big Data Transfer

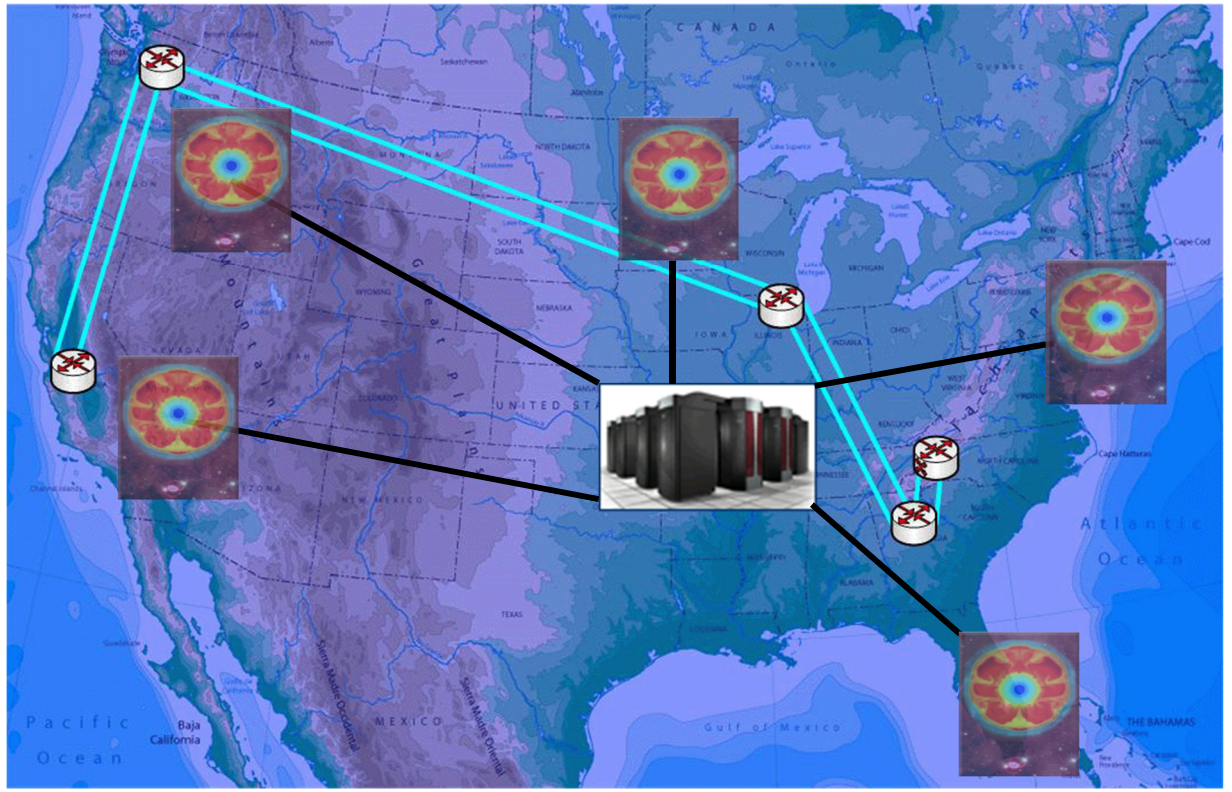
Introduction

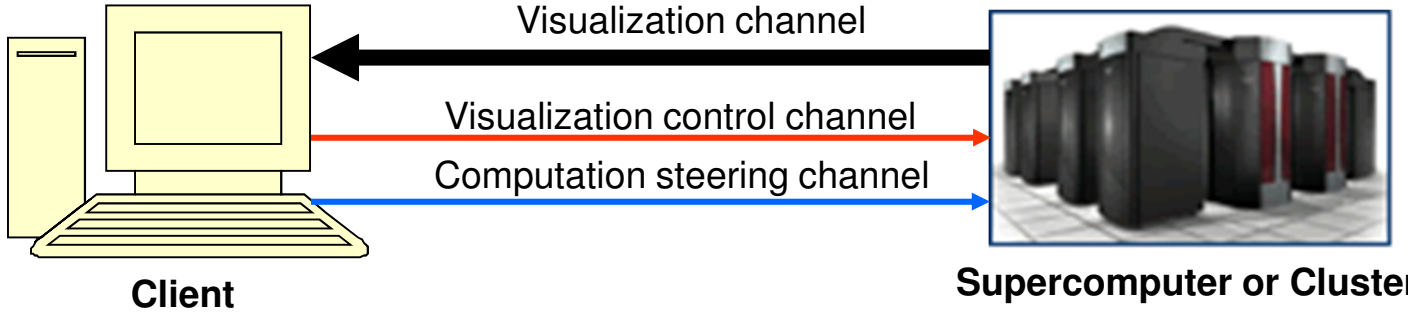
Supercomputing for big-data science



Terascale Supernova Initiative (TSI)

- Collaborative project
  - Supernova explosion
- TSI simulation
  - 1 terabyte a day with a small portion of parameters
  - From TSI to PSI
- Transfer to remote sites
  - Interactive distributed visualization
  - Collaborative data analysis
  - Computation monitoring
  - Computation steering





Challenges for Extreme-scale Scientific Applications

- A typical way of research conduct
  - Run simulation code on supercomputer in a batch mode
    - ➢ One day: 1 terabyte datasets
  - Move datasets to HPSS
    - ➢ About 8 hours
  - Transfer datasets to remote sites over the Internet
    - ➢ TCP-based transfer tools: up to one week
  - Filter out data of interest
  - Partition dataset for parallel processing
  - Extract geometry data
  - Generate images on rendering engine
  - Display results on desktop, laptop, powerwall, etc.

Visualization:

several hours or days

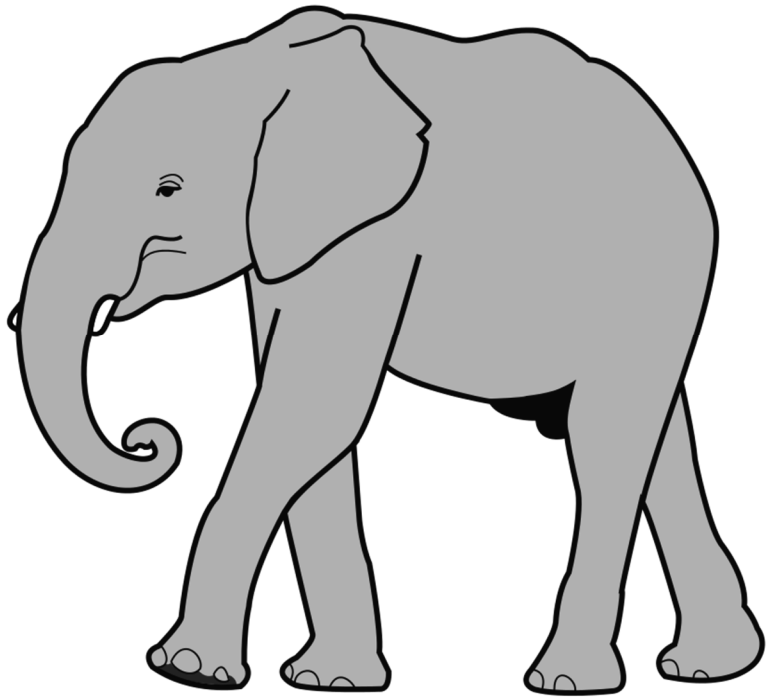
Start over if any parameter values are not set appropriately!

Challenges in Modern Sciences

- From T(erabyte) to P(etabyte), to E(xabyte), to Z(ettabyte), to Y(ottabyte), and beyond…
- Sciences: Simulation, Experimental, Observational
- Business: Financial Transactions
- Social Media: Weblogs, Twitter feeds, etc.

No matter which type of data is considered, we need a high-performance end-to-end computing solution to support data generation, storage, transfer, processing, and analysis!

V.S.



Big-data Scientific Workflows

- Require massively distributed resources
  - **Hardware**: computing facilities, storage systems, special rendering engines, display devices (tiled display, powerwall, etc.), network infrastructures, etc.
  - **Software**: Domain-specific data analytics/processing tools, programs, etc.
  - **Data**: Real-time, archival

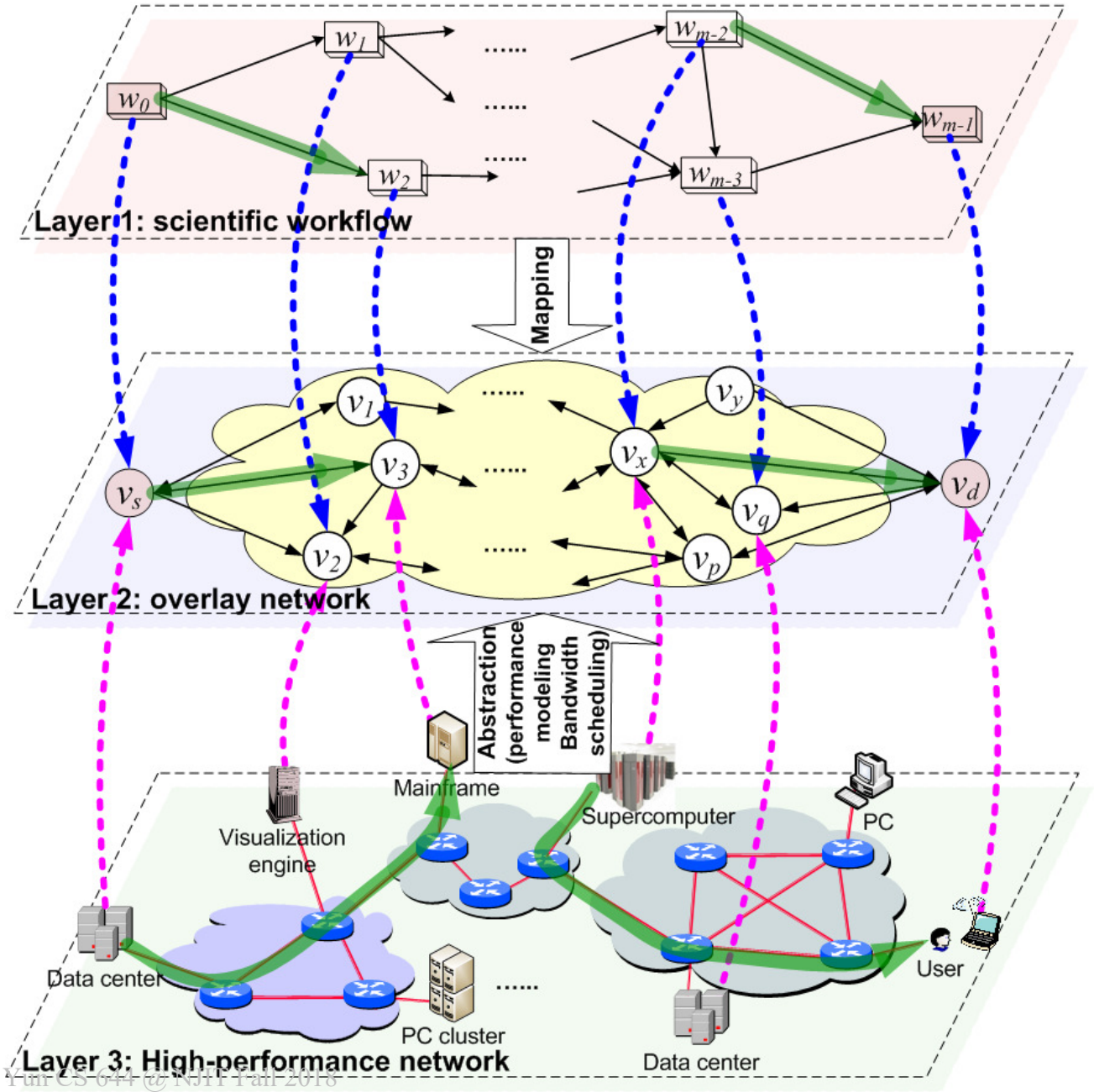
- Feature different complexities
  - **Simple case**: linear pipeline (a special case of DAG)
  - **Complex case**: DAG-structured graph

- Different application types have different performance requirements
  - **Interactive**: minimize total end-to-end delay for fast response
  - **Streaming**: maximize frame rate to achieve smooth data flow

Ultimate Goals

- Support distributed workflows in heterogeneous environments
- Optimize workflow performance to meet various user requirements
  - Delay, throughput, reliability, etc.
  - Remote visualization, online computational monitoring and steering, etc.
- Make the best use of computing and networking resources

Solution: A Three-layer Architecture



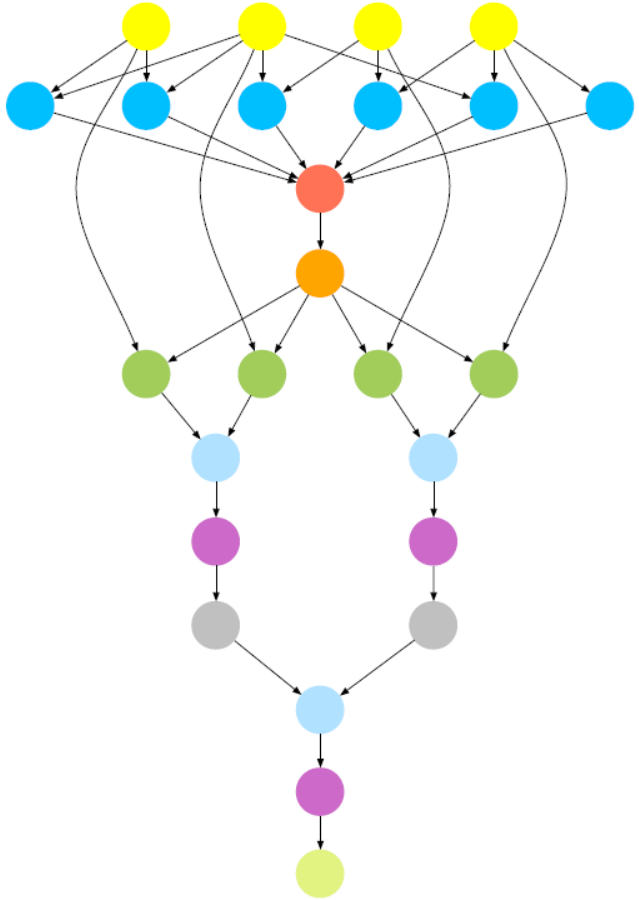
Enabling Technologies

- **Three layers**
  - Top: Abstract scientific workflow
  - Middle: Virtual overlay network (grid, cloud)
  - Bottom: Physical high-performance network
- **Top and bottom layers meet at middle layer**
  - From bottom to middle: resource abstraction
    - Bandwidth scheduling
    - Performance modeling and prediction
  - From top to middle: workflow mapping
    - Optimization: where to execute modules?
- **Workflow execution**
  - Actual data transfer: transport control
  - Actual module running: job scheduling

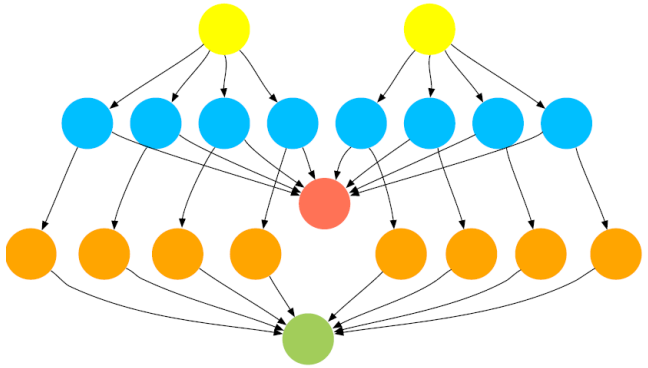
Computing for Big Data

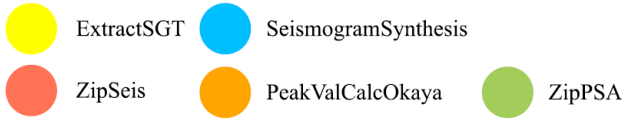
Workflow Management and Optimization

Example Scientific Workflows

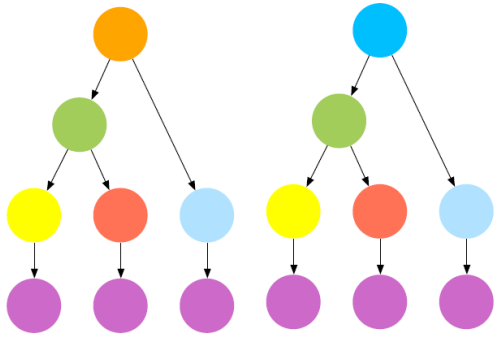


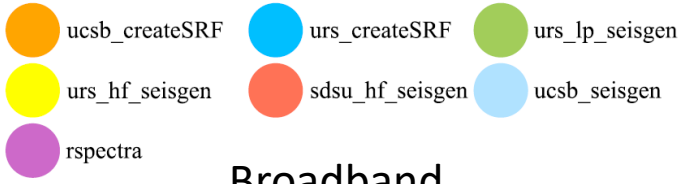
Montage



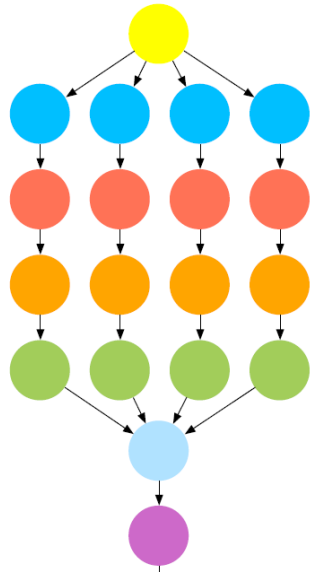


CyberShake

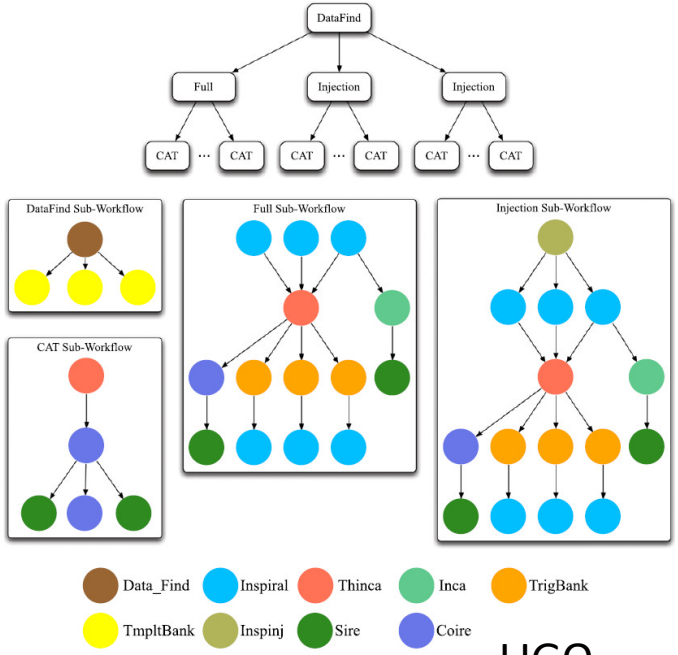




Broadband



Epigenomics

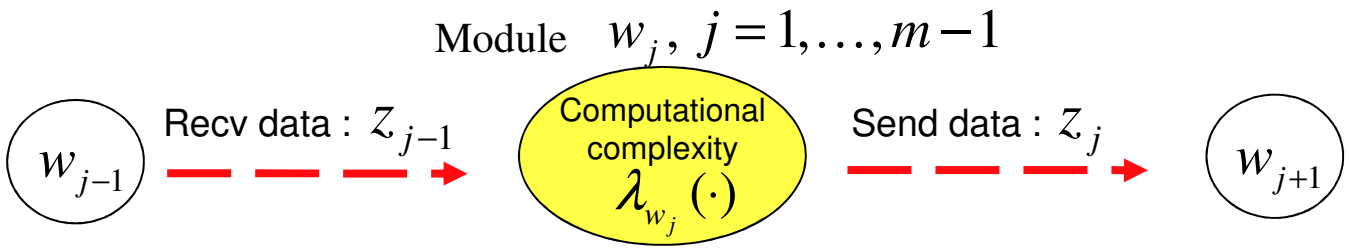


LIGO

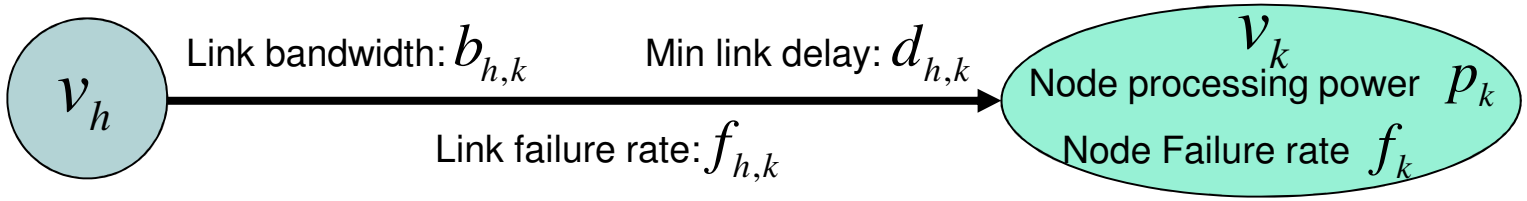
Juve et al. Characterizing and profiling scientific workflows. FGCS 2013.

Cost Models

- Computing workflow



- Computer network



$$\text{Message transmission time}: \frac{z_{i,j}}{b_{i,j}} + d_{i,j}, \quad \text{Node computing time}: \frac{\lambda_{w_i}(z_{w_i})}{p}$$

- Objectives: map modules to nodes to achieve minimum end-to-end delay (MED) or maximum frame rate (MFR)

Workflow mapping and execution conditions

- Single-node mapping
  - If w is mapped to v, $x_{wv} = 1$
  - Otherwise, $x_{wv} = 0$

  } s.t. $\sum_{v \in V_c} x_{wv} = 1, \forall w \in V_w$

- Module execution dependence

$$t^s_{w_j} \geq t^f_{e_{i,j}}, \forall w_j \in V_w, e_{i,j} \in E_w$$

- Data transfer precedence

$$t^s_{e_{i,j}} \geq t^f_{w_i}, \forall w_i \in V_w, e_{i,j} \in E_w$$

Workflow mapping and execution conditions

- Fair node sharing

$$T_{\text{comp}}(w,v) = \frac{\lambda_w(z_w)\cdot A(w,v)}{p_v}, \quad \forall w \in V_w, v \in V_c$$

where $A(w,v) = \int_{t_w^s}^{t_w^f} \alpha_v(t)dt$ and $\alpha_v(t) = \sum_{w \in V_w : (t_w^f - t)(t - t_w^s) \geq 0} x_{wv}$

- Fair link sharing

$$T_{\text{tran}}(e_{i,j}, l_{h,k}) = \frac{z_{i,j} \cdot B(e_{i,j}, l_{h,k})}{b_{h,k}} + d_{h,k}, \quad \forall e_{i,j} \in E_w, l_{h,k} \in E_c$$

where $B(e_{i,j}, l_{h,k}) = \int_{t_{e_{i,j}}^s}^{t_{e_{i,j}}^f} \beta_{l_{h,k}}(t)dt$ and $\beta_{l_{h,k}}(t) = \sum_{e_{i,j} \in E_w : (t_{e_{i,j}}^f - t)(t - t_{e_{i,j}}^s) \geq 0} x_{w_i v_h} \cdot x_{w_j v_k}$

Performance Metrics

- End-to-end Delay (ED)

$$T_{\text{ED}}(\text{CP mapped to a path } P \text{ of } q \text{ nodes})$$

$$= T_{\text{comp}} + T_{\text{tran}} = \sum_{i=0}^{q-1} T_{g_i} + \sum_{i=0}^{q-2} T_{e(g_i, g_{i+1})}$$

$$= \sum_{i=0}^{q-1} \left(\sum_{j \in g_i,\, j \geq 1} \frac{\lambda_{w_j}(z_{w_j}) \cdot A(w_j, v_{P[i]})}{p_{P[i]}} \right)$$

$$+ \sum_{i=0}^{q-2} \left(\frac{z(g_i, g_{i+1}) \cdot B(e(g_i, g_{i+1}), l_{P[i],P[i+1]})}{b_{P[i],P[i+1]}} + d_{P[i],P[i+1]} \right)$$

Performance Metrics

- Frame rate: inverse of global bottleneck (BN)

$$T_{\text{BN}}(G_w \text{ mapped to a network } G_c)$$

$$= \max_{\substack{w_i \in V_w, e_{j,k} \in E_w \\ v_{i'} \in V_c, l_{j',k'} \in Ec}} \begin{pmatrix} T_{\text{comp}}(w_i, v_{i'}) \\ T_{\text{tran}}(e_{j,k}, l_{j',k'}) \end{pmatrix} = \max_{\substack{w_i \in V_w, e_{j,k} \in E_w \\ v_{i'} \in V_c, l_{j',k'} \in Ec}} \begin{pmatrix} \dfrac{\lambda_{w_i}(z_{w_i}) \cdot A(w_i, v_{i'})}{p_{i'}} \\ \dfrac{z_{j,k} \cdot B(e_{j,k}, l_{j',k'})}{b_{j',k'}} + d_{j',k'} \end{pmatrix}$$

- Overall failure rate

$$\mathscr{F} = 1 - \left(\prod_{\substack{e_{i,j} \text{ mapped on } l_{h,k} \\ e_{i,j} \in E_w, l_{h,k} \in E_c}} (1 - f_{h,k}) \right) \cdot \left(\prod_{\substack{w_i \text{ mapped on } v_h \\ w_i \in V_w, v_h \in V_c}} (1 - f_h) \right)$$

Objective Functions

- Minimum End-to-end Delay (MED)

$$\min_{\text{all possible mappings}} (T_{\text{ED}}), \quad \text{such that} \quad \mathscr{F} \leq \mathbb{F}$$

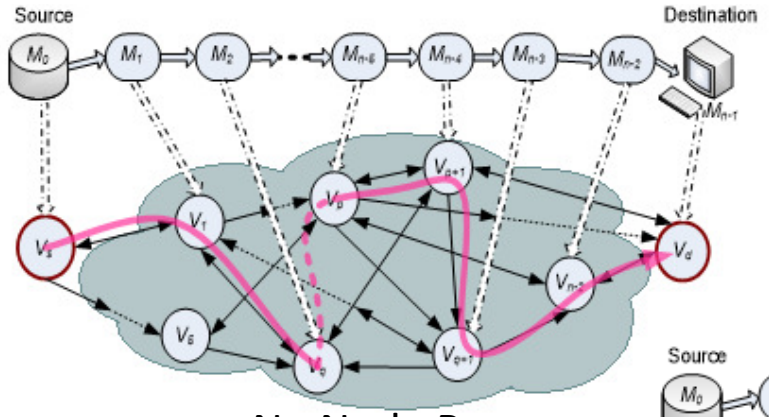
- Maximum Frame Rate (MFR)

$$\min_{\text{all possible mappings}} (T_{\text{BN}}), \quad \text{such that} \quad \mathscr{F} \leq \mathbb{F}$$

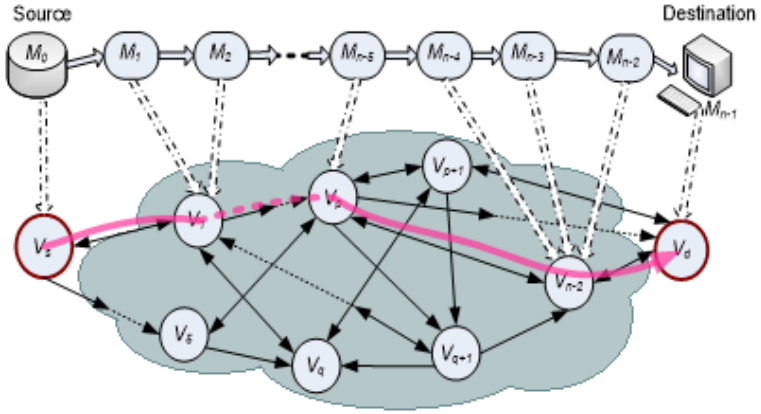
Pipeline Mapping – A special case of DAG

Problem categories

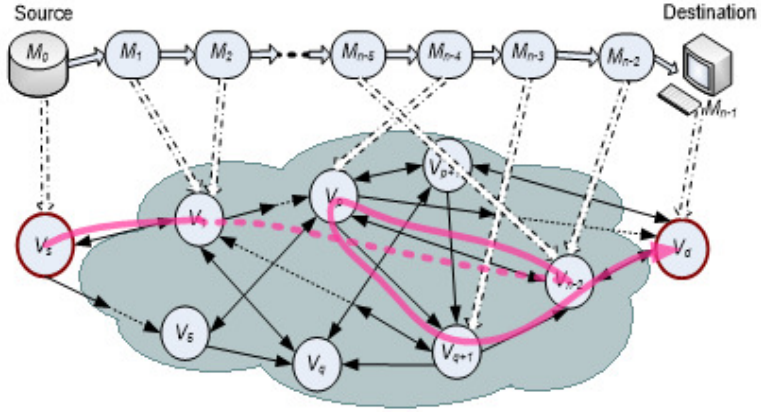
- MED
  - No Node Reuse (MED-NNR)
  - Contiguous Node Reuse (MED-CNR)
  - Arbitrary Node Reuse (MED-ANR)
- MFR
  - No Node Reuse of Share (MFR-NNR)
  - Contiguous Node Reuse or Share (MFR-CNR)
  - Arbitrary Node Reuse or Share (MFR-ANR)



No Node Reuse



Contiguous Node Reuse



Arbitrary Node Reuse

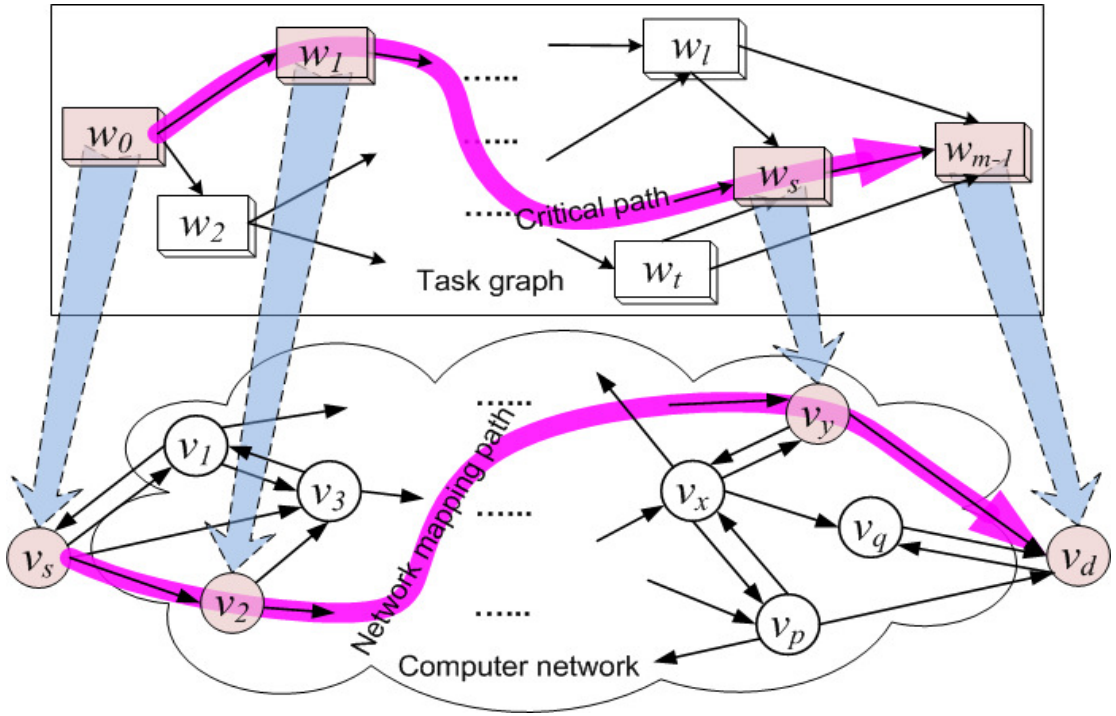
Problem Category and Complexity

Objective Function / Constraints	Minimum End-to-end Delay	Maximum Frame Rate
No Node Reuse	NP-complete	NP-complete
Contiguous Node Reuse	NP-complete	NP-complete
Arbitrary Node Reuse	Polynomial (Dyn. Prog.)	NP-complete

- MED-ANR is solvable in polynomial time
  - Dynamic Programming -based solution
- MED/MFR-NNR/CNR are NP-complete
  - Reduce from DISJOINT-CONNECTING-PATH (DCP)
- MFR-ANR is NP-complete
  - Reduce from Widest-path with Linear Capacity Constraints (WLCC)

From special to general: DAG Mapping

- Mapping algorithm for MED
  - Recursive Critical Path for MED with Failure Rate Constraint



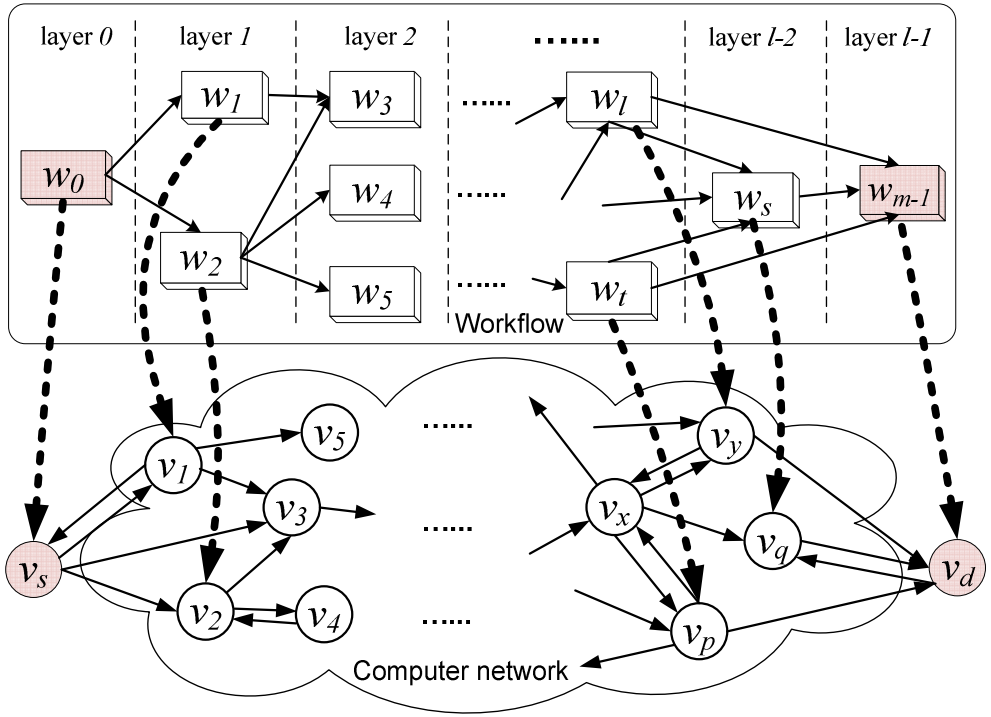
Gu et al. Performance Analysis and Optimization of Distributed Computing Workflows in Heterogeneous Network Environments. IEEE Trans. on Computers, 2016.

From special to general: DAG Mapping

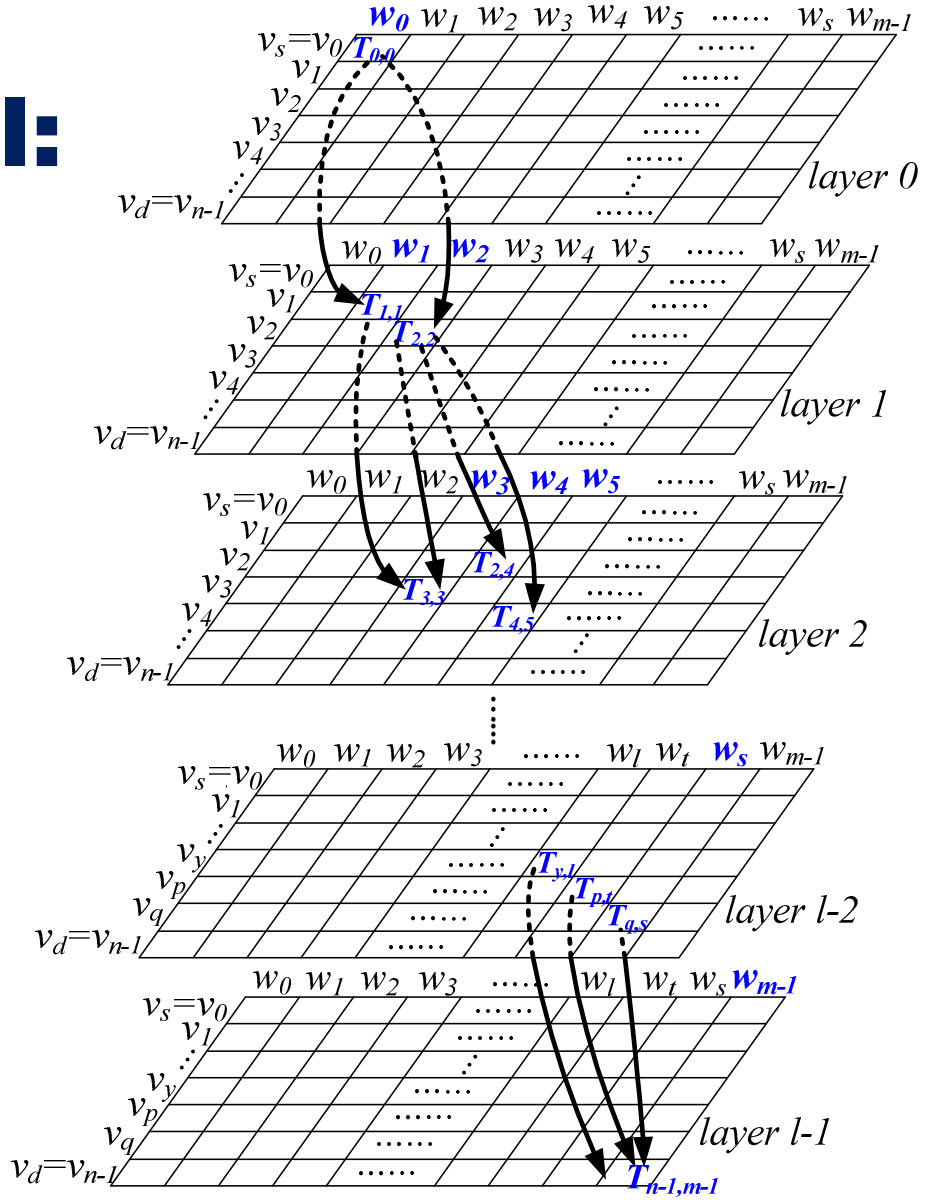
- Mapping algorithm for MFR
  - Layer-oriented DP algorithm for MFR with Failure Rate Constraint
    - Sort a DAG-structure workflow in a topological order
    - Map computing modules to network nodes on a layer-by-layer basis, considering: i) module dependency in the workflow; ii) network connectivity; and iii) node and link failure rate

$$T_{i,j}^{\,v_i \in \bigcap_{\forall v \in V_{mapping}(pre(w_j))} (suc(v))}_{j=1 \text{ to } m-1,} = \max_{\substack{w_u \in pre(w_j), \\ 0 \le h \le n-1}} \begin{cases} \max \left(\begin{array}{l} T_{h,u} \\ \dfrac{z_{u,j} \cdot B(e_{u,j}, l_{h,i})}{b_{h,i}} + d_{h,i} \\ \dfrac{\lambda_{w_j}(z_{w_j}) \cdot A(w_j, v_i)}{p_i} \end{array} \right) \wedge \mathscr{F}_j = 1 - (1 - \mathscr{F}_u) \cdot (1 - f_{h,i}) \cdot (1 - f_i) \le \mathbb{F} & \text{if } h \ne i \\ \max \left(\begin{array}{l} T_{h,u} \\ \dfrac{\lambda_{w_j}(z_{w_j}) \cdot A(w_j, v_h)}{p_h} \end{array} \right) \wedge \mathscr{F}_j = 1 - (1 - \mathscr{F}_u) \cdot (1 - f_i) \le \mathbb{F} & \text{if } h = i \end{cases}$$

From special to general: DAG Mapping



Layered workflow in a topological sorting.



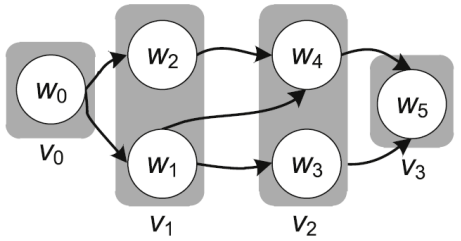
The DP table with separated layers

A further step: on-node scheduling

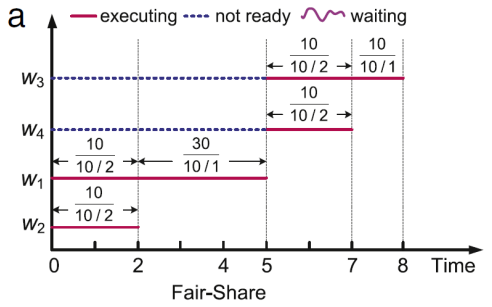
- When multiple modules are mapped to the same node
- Carefully scheduling how the node's processing power be shared may be helpful to achieve MED
- Critical Path-based Priority Scheduling algorithm

Module	CR	Mapped To (PP)
w_0	0	v_0 (---)
w_1	40	v_1 (10)
w_2	10	v_1 (10)
w_3	20	v_2 (10)
w_4	10	v_2 (10)
w_5	0	v_3 (---)

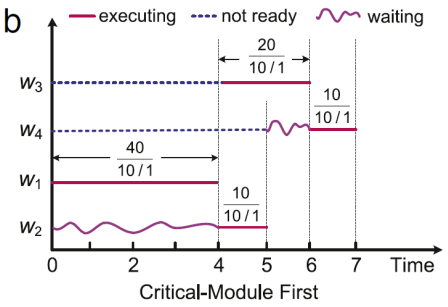
Workflow parameters



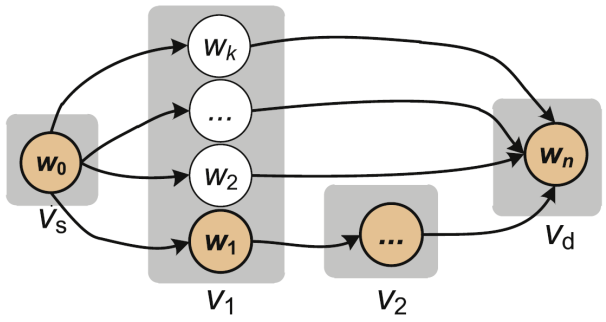
Workflow structure & mapping scheme

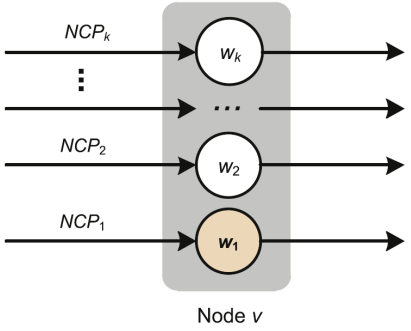


Fair-Share



Critical-Module First



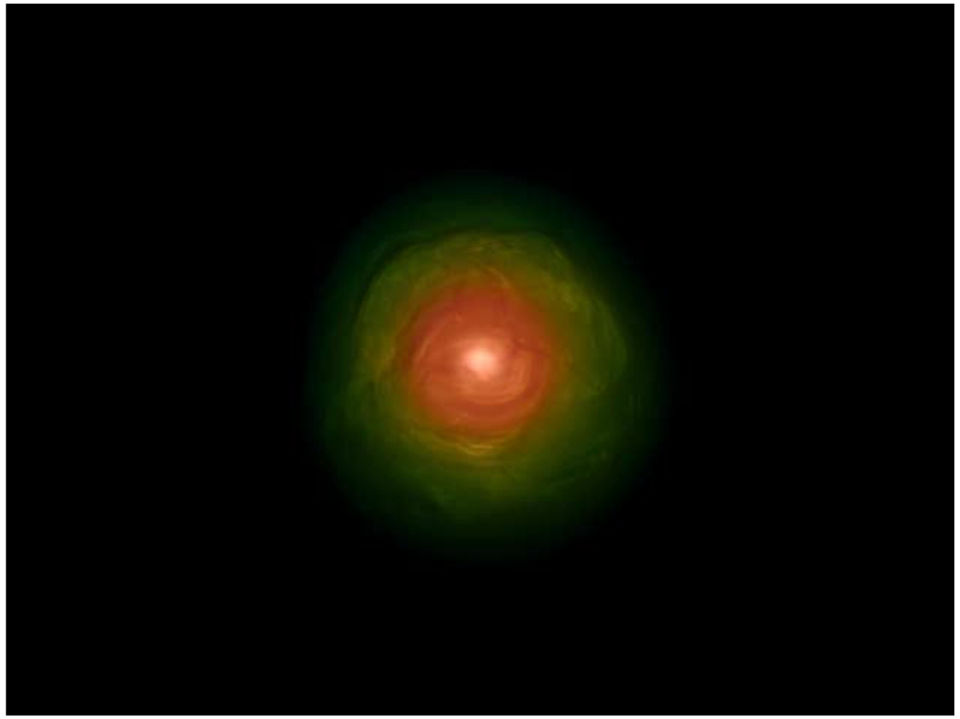


Yun et al. An Integrated Approach to Workflow Mapping and Task Scheduling for Delay Minimization in Distributed Environments. JPDC, 2015.

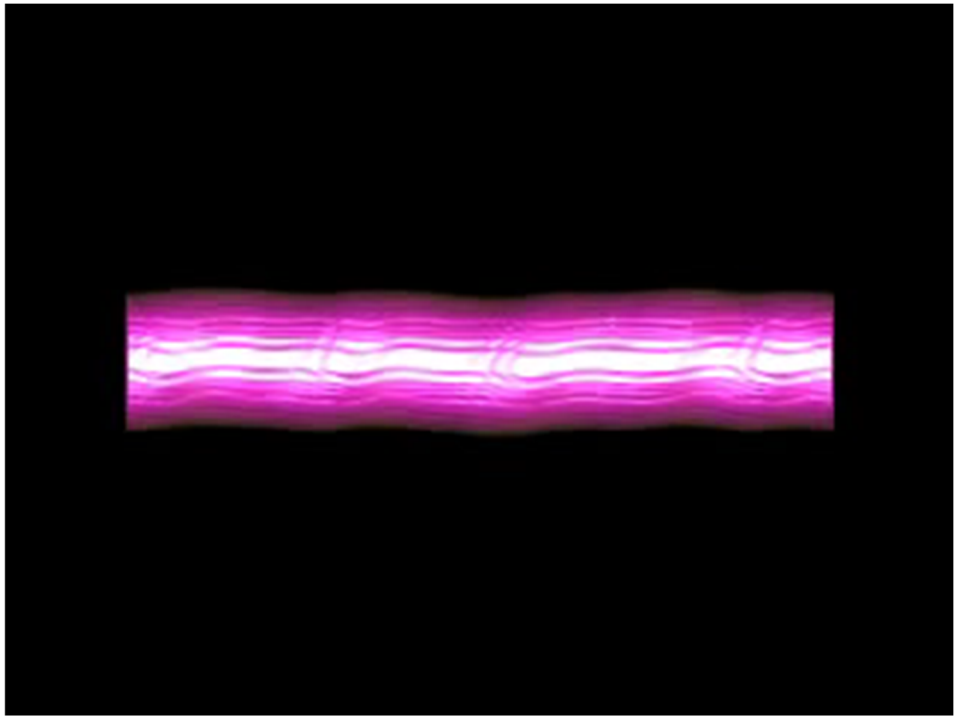
A Prototype System: Distributed Remote Intelligent Visualization Environment (DRIVE)

- Two Examples in the Visualization of Large-scale Scientific Applications

TSI explosion (density, raycasting)

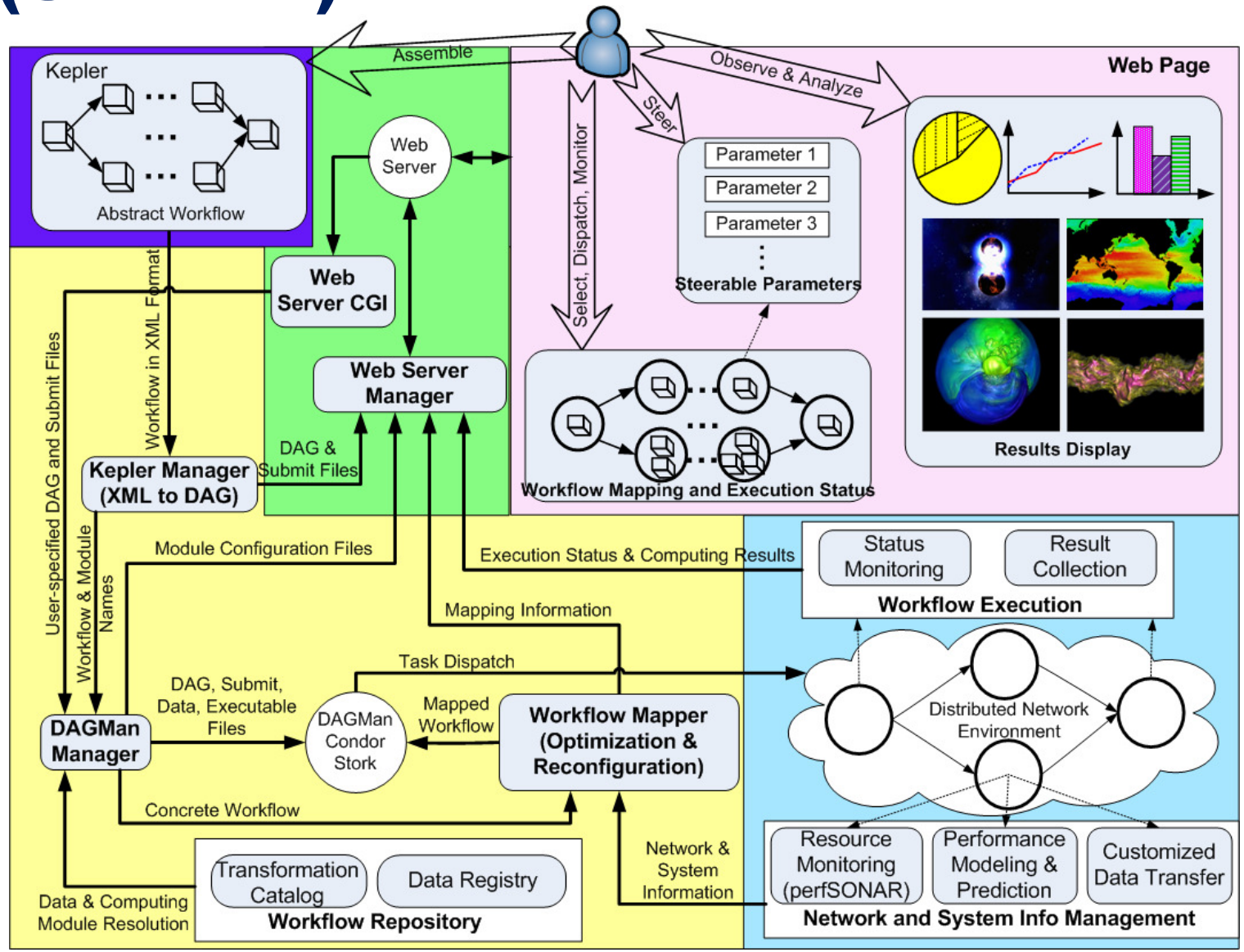


Jet air flow dynamics (pressure, raycasting)



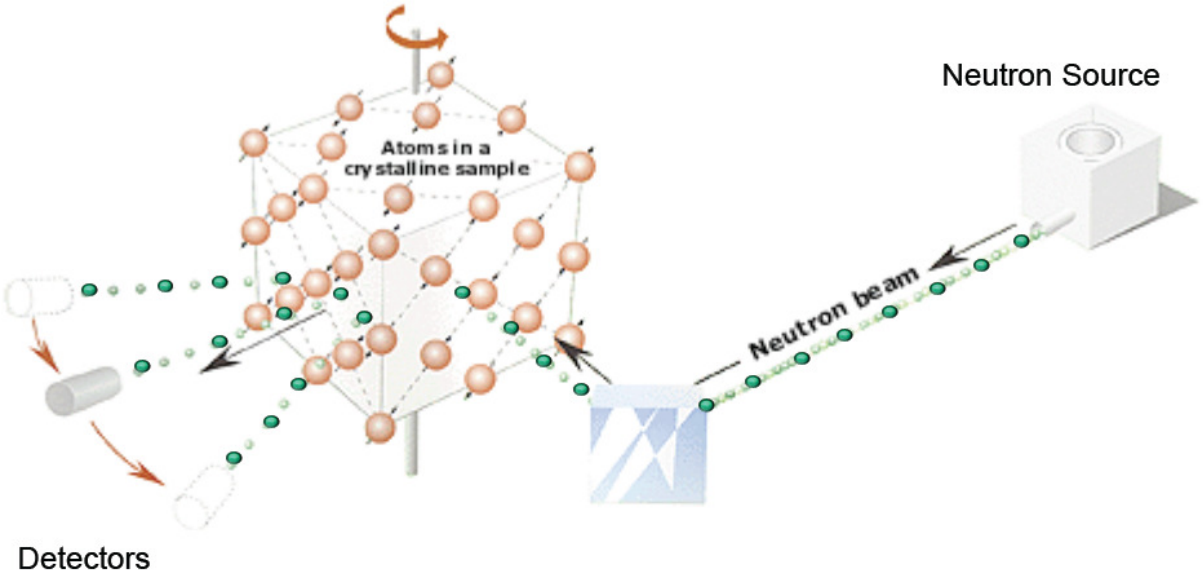
Wu et al. Self-adaptive configuration of visualization pipeline over wide-area networks. IEEE Transactions on Computers, 2008

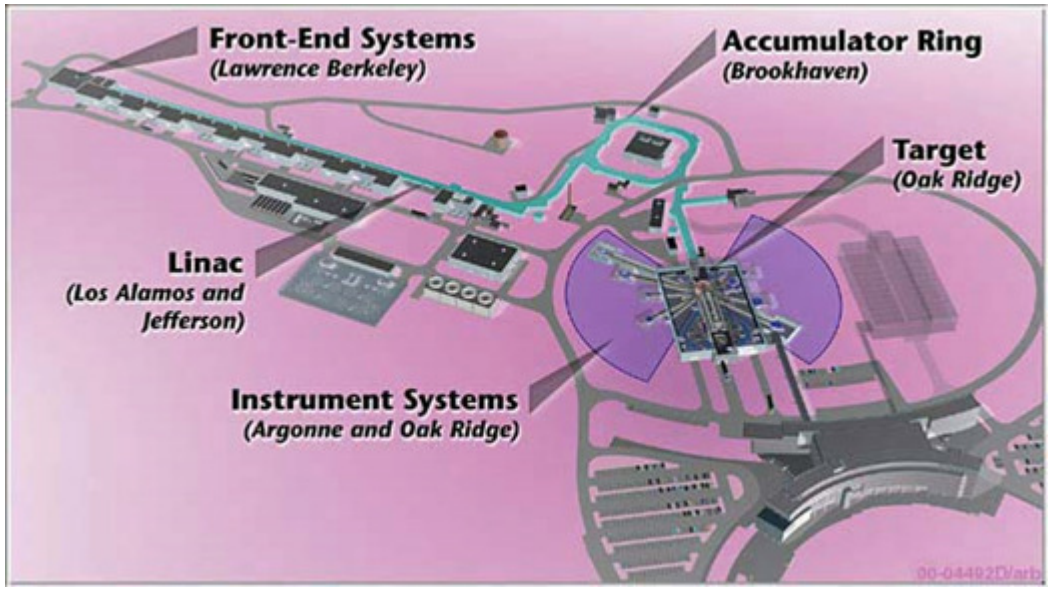
Scientific Workflow Automation and Management Platform (SWAMP)



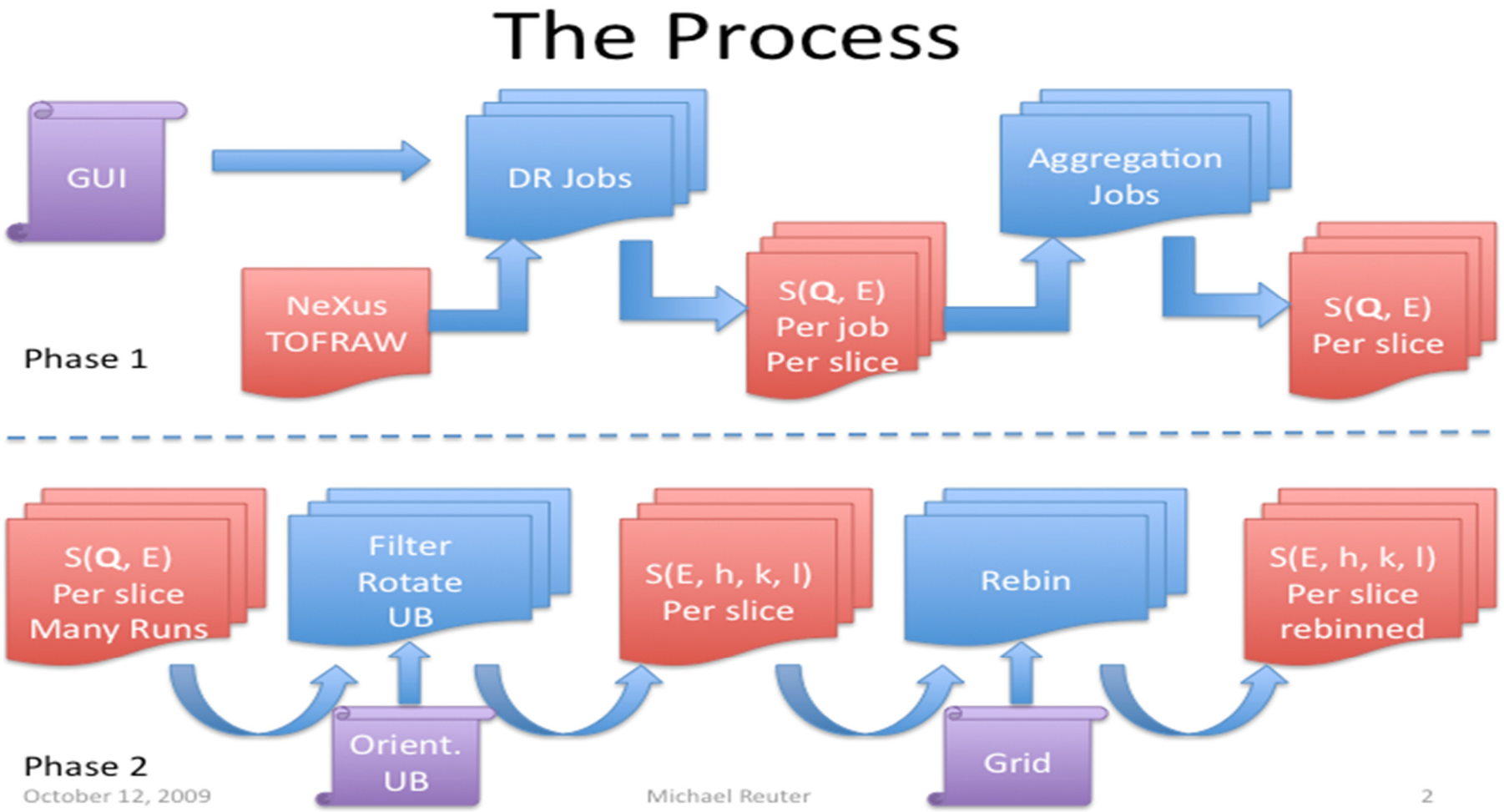
Gu and Wu. Distributed Throughput Optimization for Large-scale Scientific Workflows under Fault-tolerance Constraint. Journal of Grid Computing, 2013.

Use case: Spallation Neutron Source (SNS)

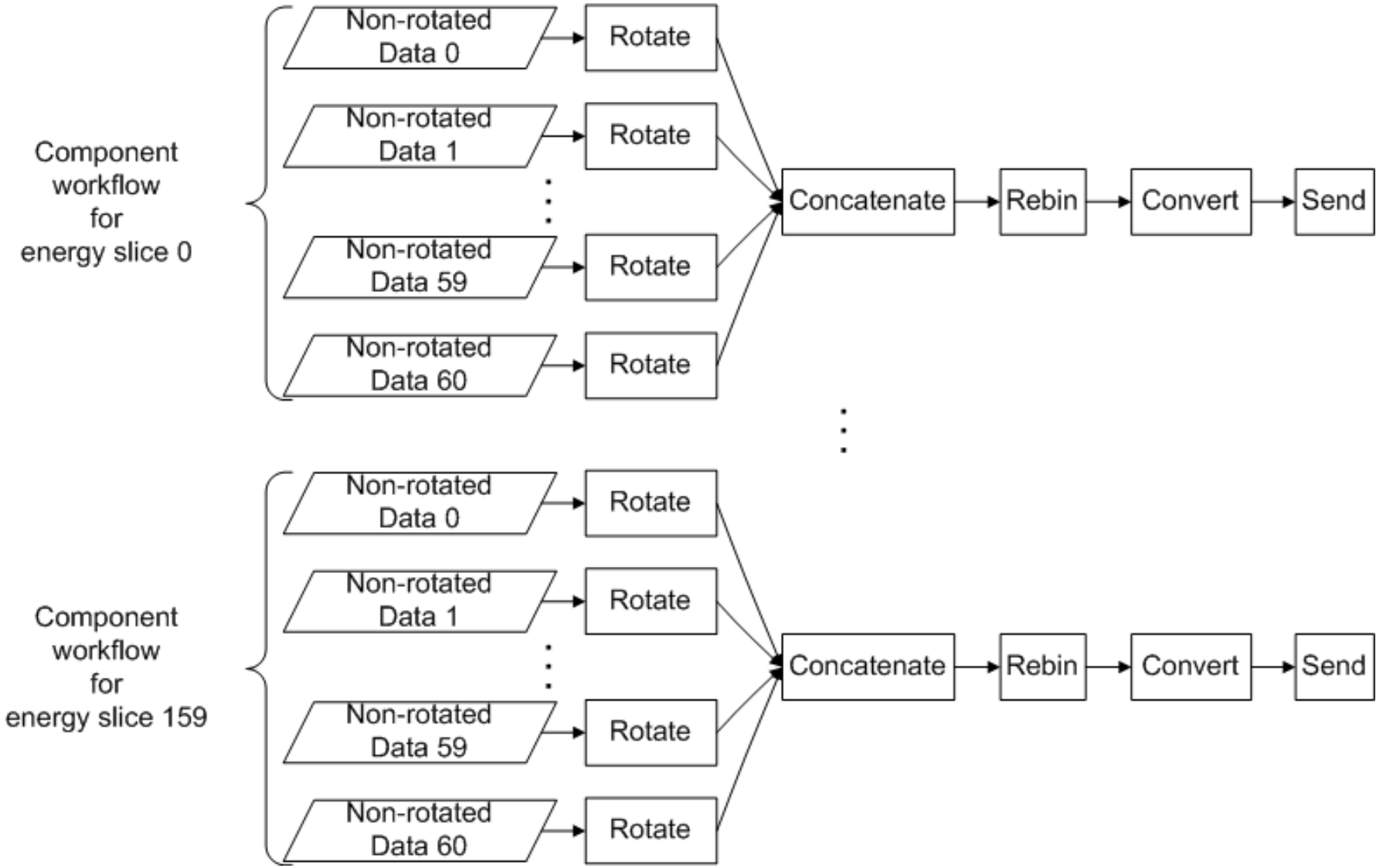




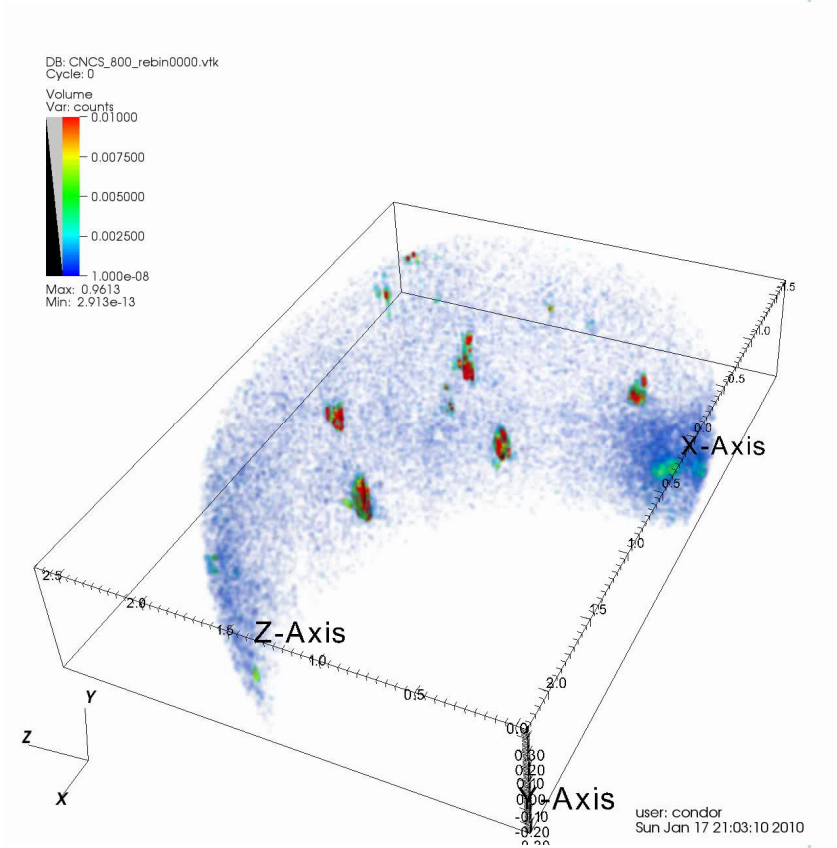
SNS Workflow (abstract)



SNS Workflow (concrete)



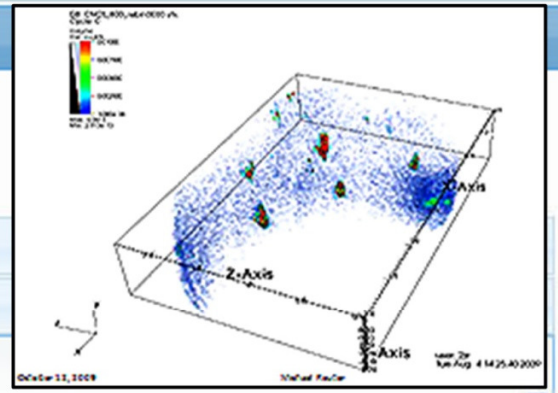
Using SWAMP



Welcome | Create | Dispatch | Results

Dispatch a Workflow

LINTEST.META.XML



- Select a Workflow
- Dispatch Workflow
- Customize Workflow

Customize this workflow by clicking on the sub-workflows below. Click the Dispatch Workflow button above when you are ready to submit.

Show 10 entries Search:

	Workflow Name	Details	Version
☐	0	Topology	default
☐	1	Topology	default
☐	2	Topology	default
☐	3	Topology	default
☐	4	Topology	default
☐	5	Topology	default
☐	6	Topology	default
☐	7	Topology	default
☐	8	Topology	default
☐	9	Topology	default

Select All | Select None | Remove All Customization

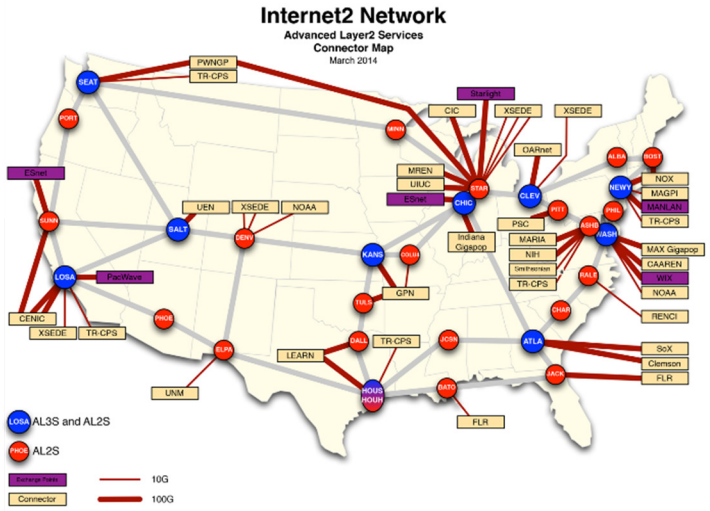
Showing 1 to 10 of 160 entries First | Previous | 1 | 2 | 3 | 4 | 5 | Next | Last

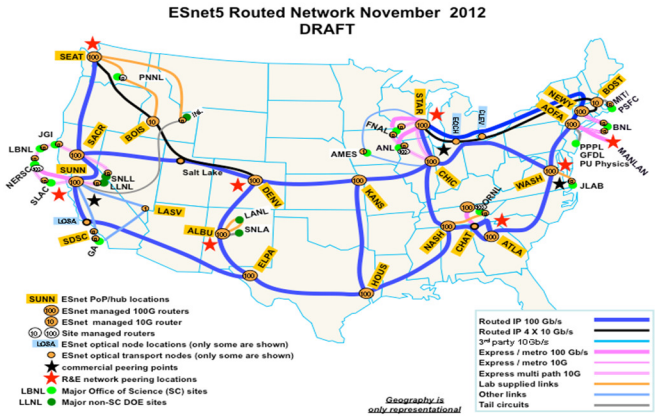
Networking for Big Data

Software-defined Networking (SDN)
High-performance Networking (HPN)

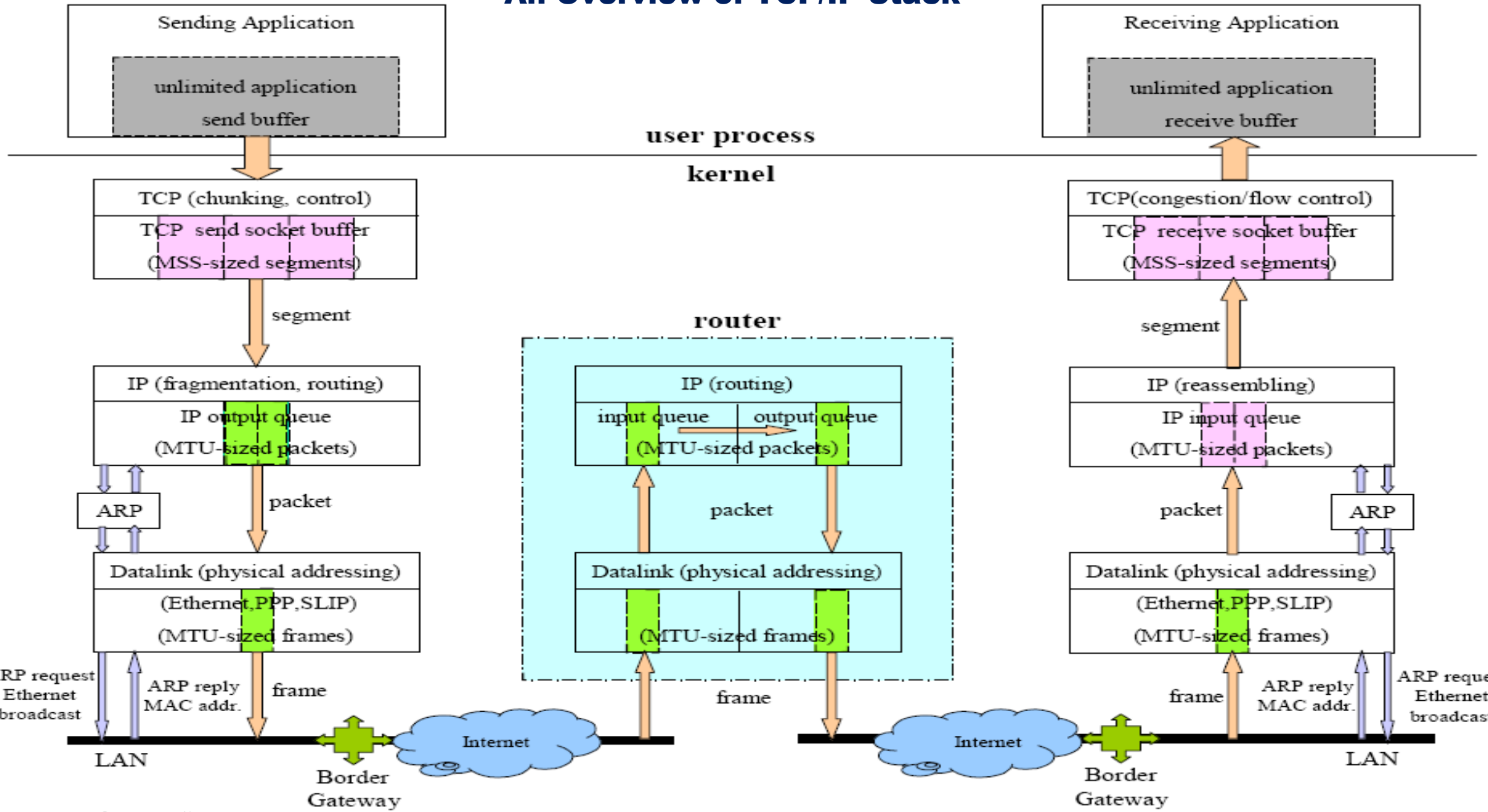
Networking Requirements

- Provision dedicated channels to meet different transport objectives
  - High bandwidths
    - Multiples of 10Gbps to terabits networking
    - Support bulk data transfers
  - Stable bandwidths
    - 100s of Mbps
    - Support interactive control operations
- Why not the Internet?
  - Only backbone has high bandwidths (last mile)
  - Packet-level resource sharing
  - Best-effort IP routing
  - TCP: hard to sustain 10s Gbps or to stabilize





An Overview of TCP/IP Stack



Software-Defined Networking

- The Concept of Virtualization
- Virtualization of Computing
- Virtualization of Networking
- Software-Defined Network
- Possible Directions

Concept of Virtualization

- Decoupling HW/SW
- Abstraction and layering
- Using, demanding, but not owning or configuring
- Resource pool: flexible to slice, resize, combine, and distribute
- A degree of automation by software

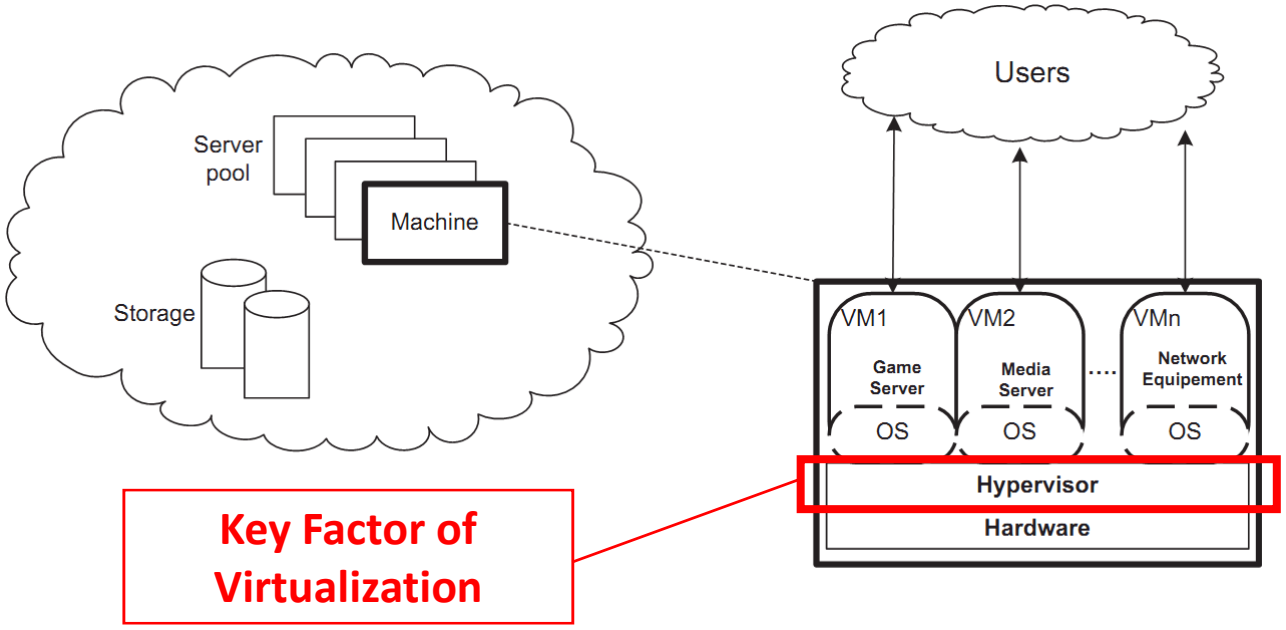
Benefits of Virtualization

- An analogy: owning a huge house
  - Real estate, immovable property
  - Does not generate cash and income

- How to gain more profit?
  - Divide this huge house into suites, and RENT to people!
  - Renting suites: using but not owning
  - Transform a static investment into cash generators!!!

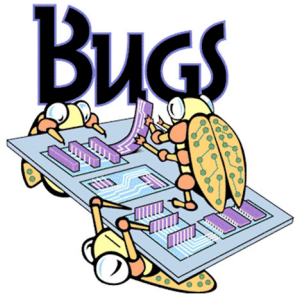
Virtualization of Computing

- Partitioning one physical machine
  - Virtual instances
    - Running concurrently, sharing resources
- Hypervisor: Virtual Machine Monitor (VMM)
  - A software layer presents abstraction of physical resources



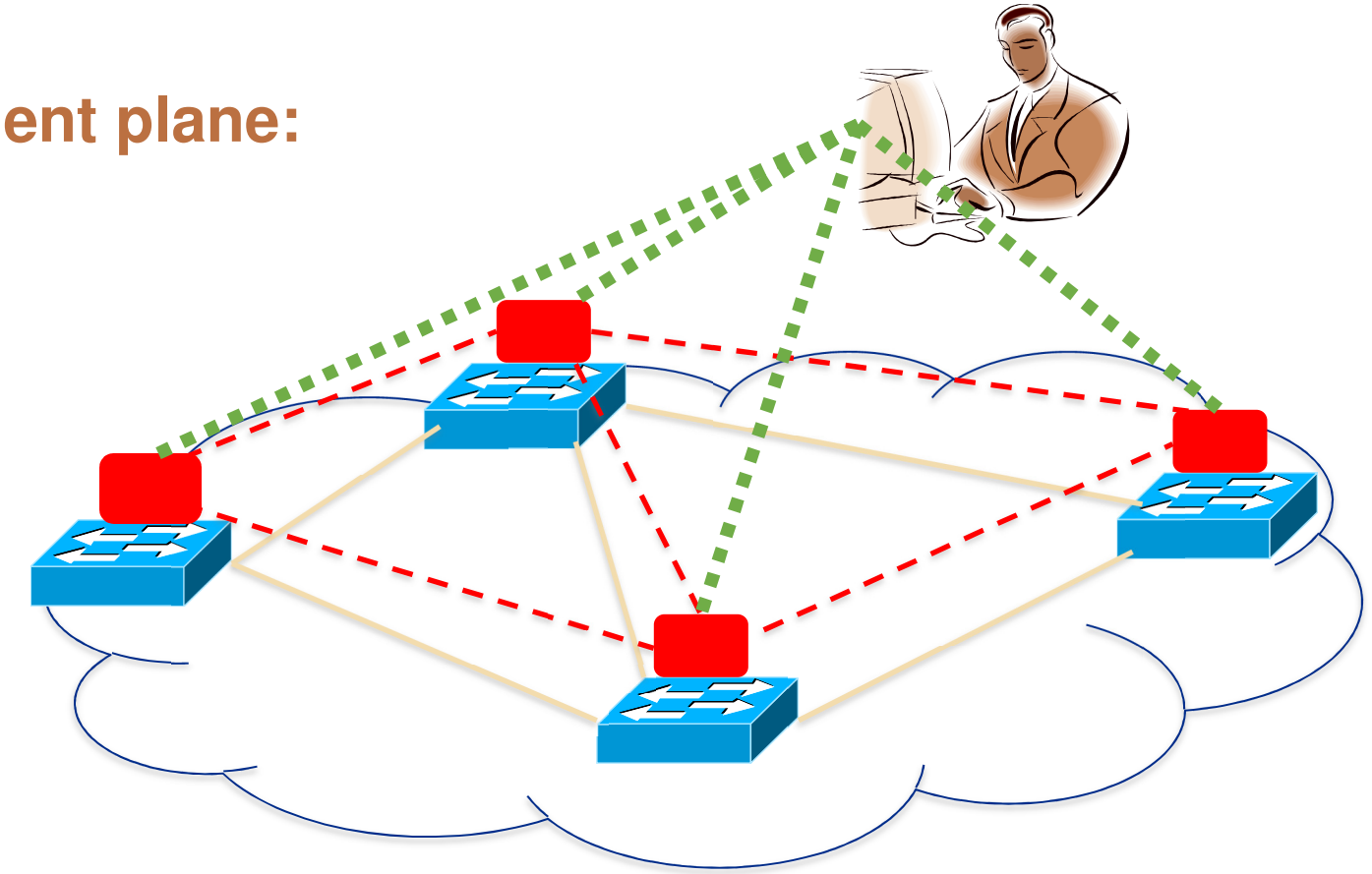
Networks are Hard to Manage

- Operating a network is expensive
  - More than half the cost of a network
  - Yet, operator error causes most outages
- Buggy software in the equipment
  - Routers with 20+ million lines of code
  - Cascading failures, vulnerabilities, etc.
- The network is “in the way”
  - Especially a problem in data centers
  - … and home networks



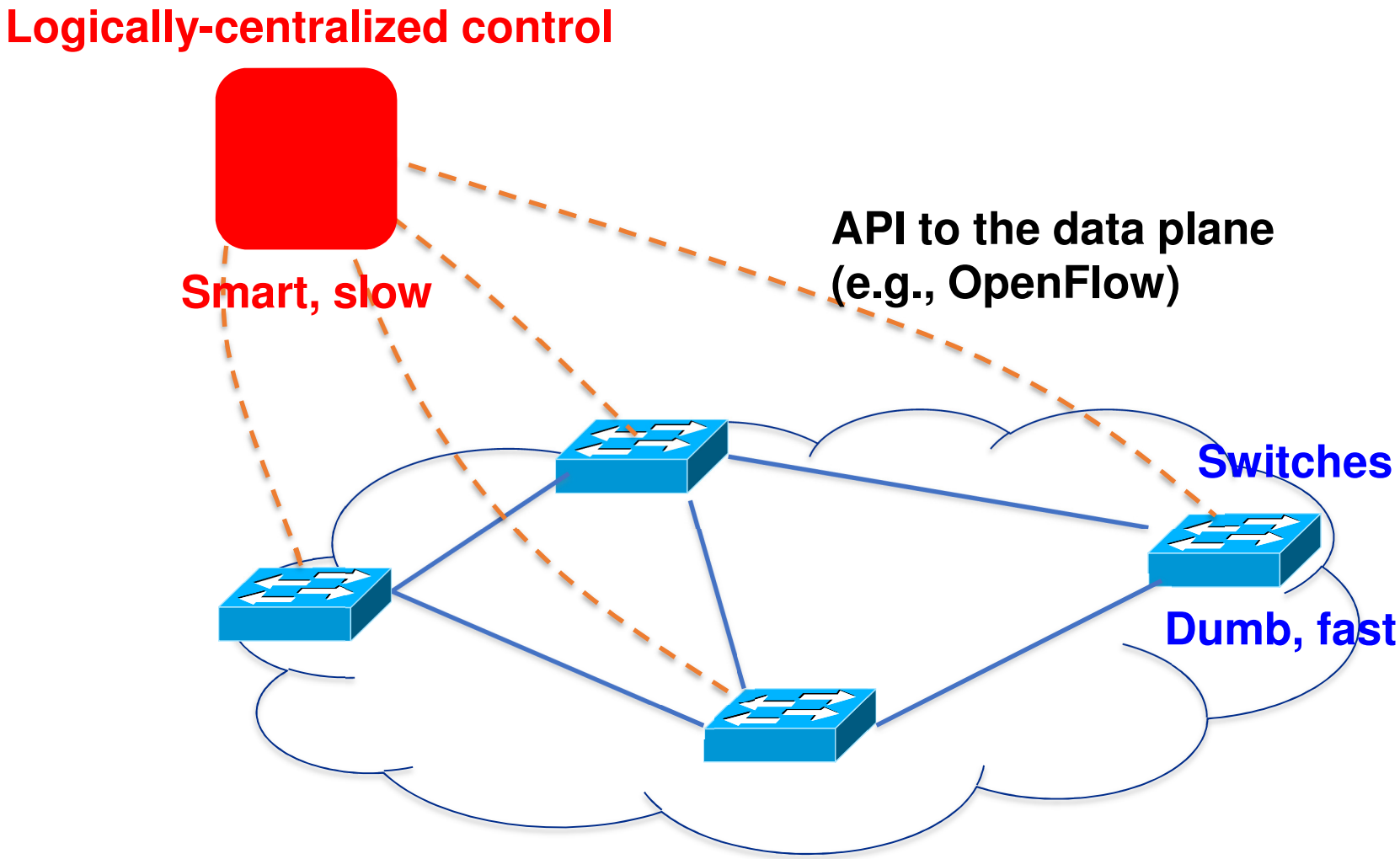
Traditional Computer Networks

Management plane:

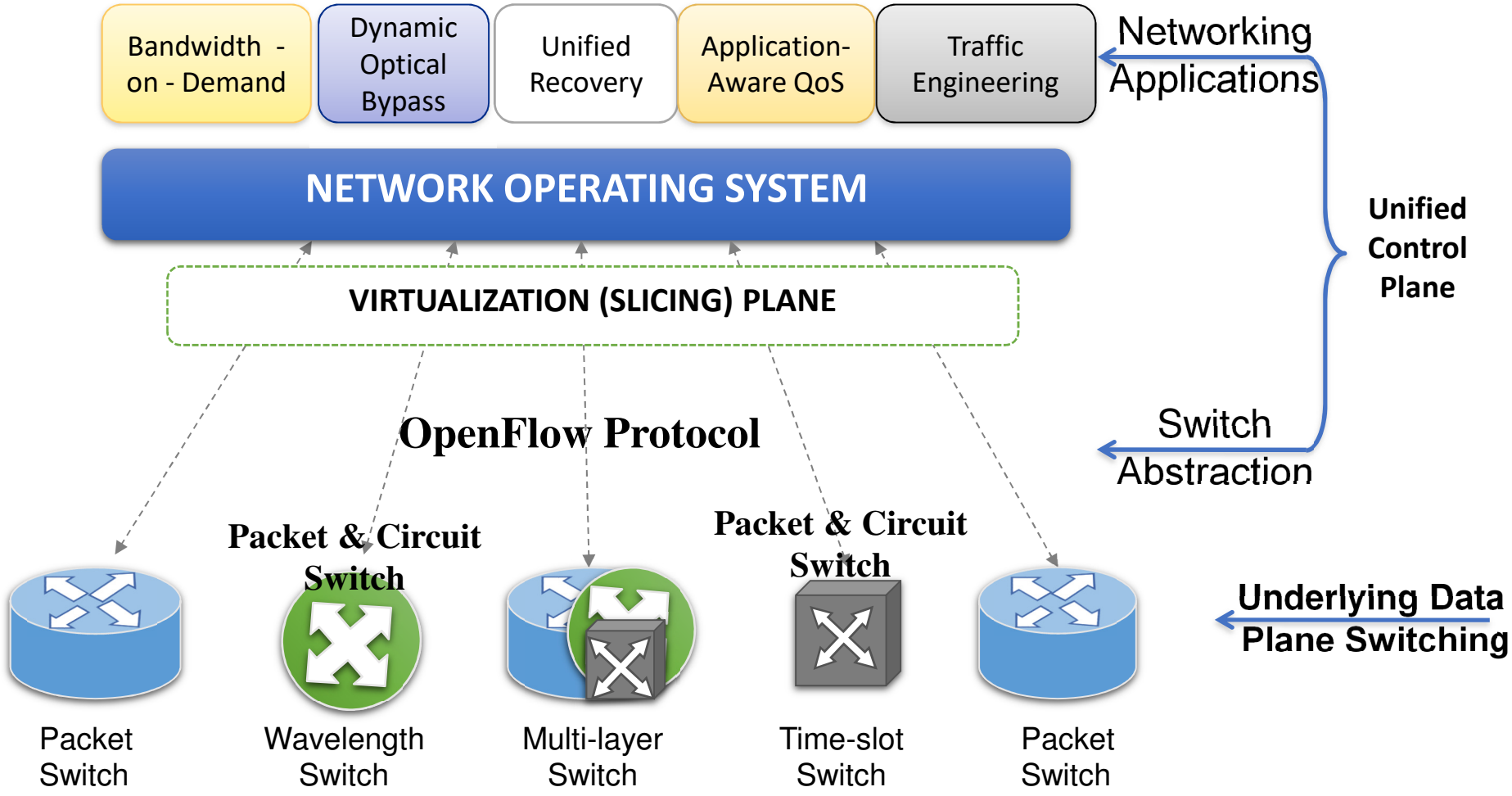


Collect measurements and configure the equipment

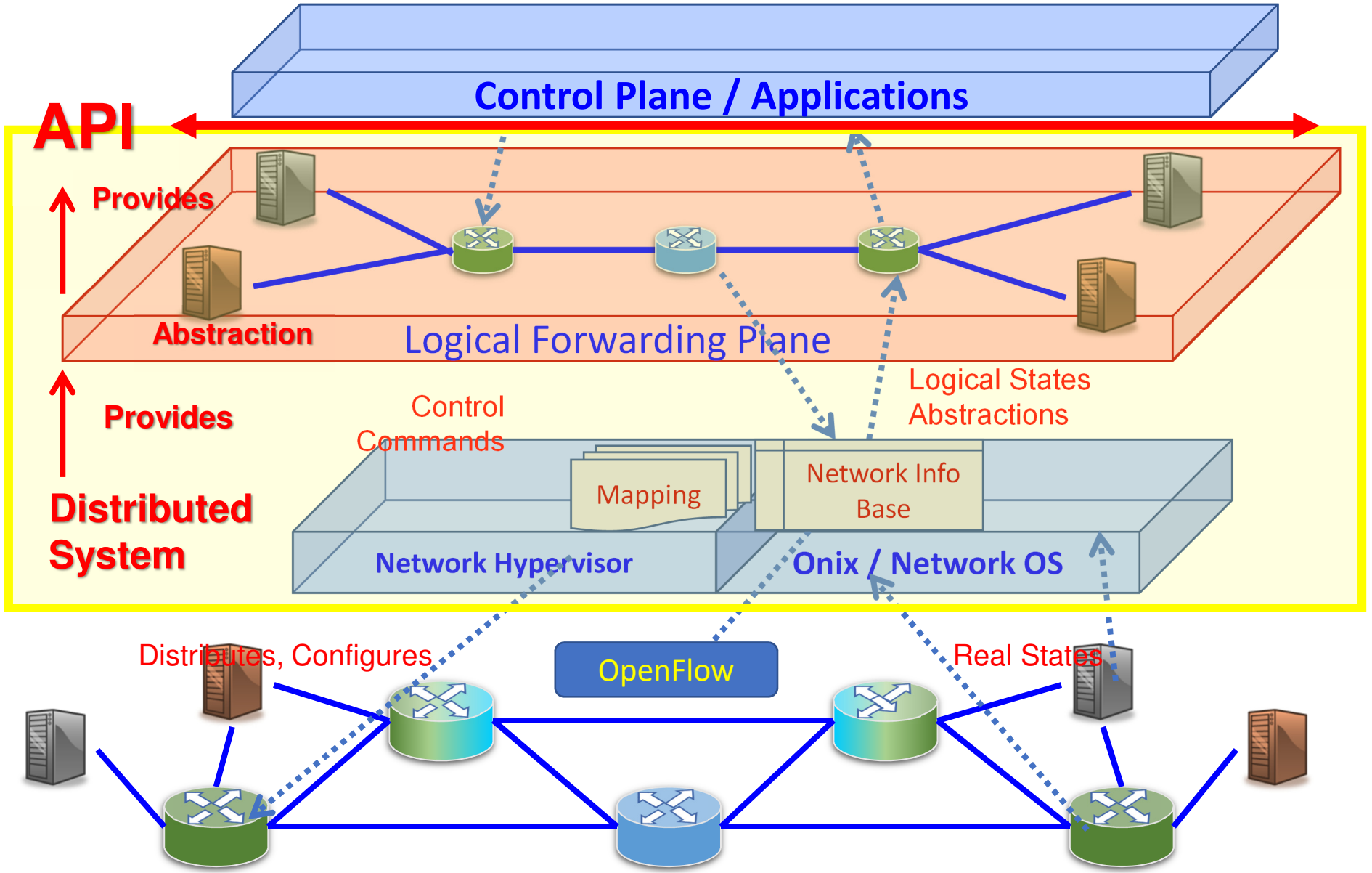
Software Defined Networking (SDN)



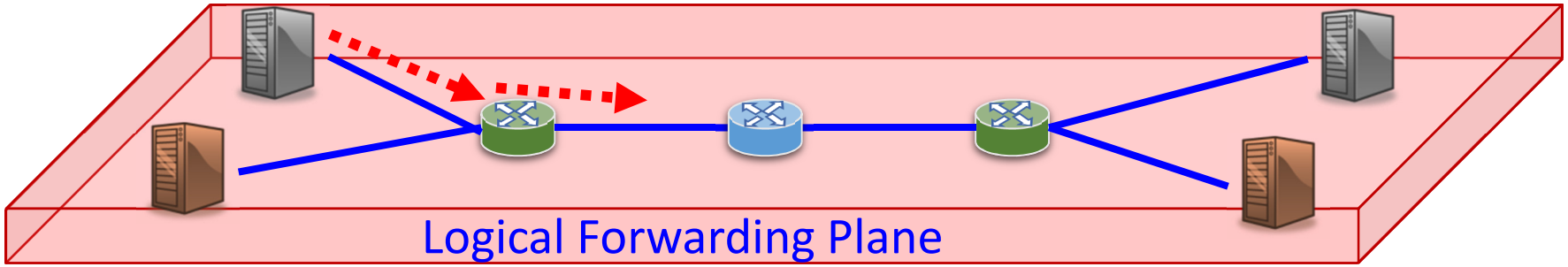
Provide Choices



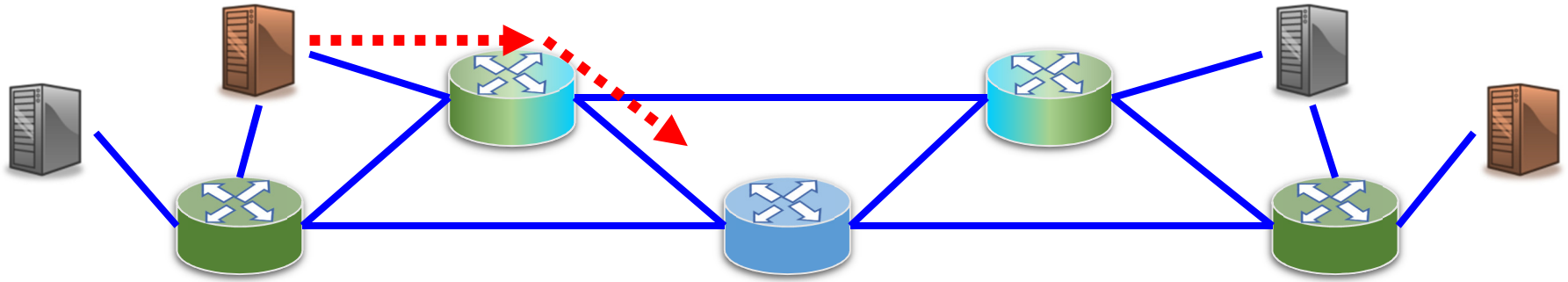
Architecture



Switch Forwarding Pipeline



As packets/flows traverse the network: moving both in **logical** and **physical** forwarding plane → **logical context**



Data-Plane: Simple Packet Handling

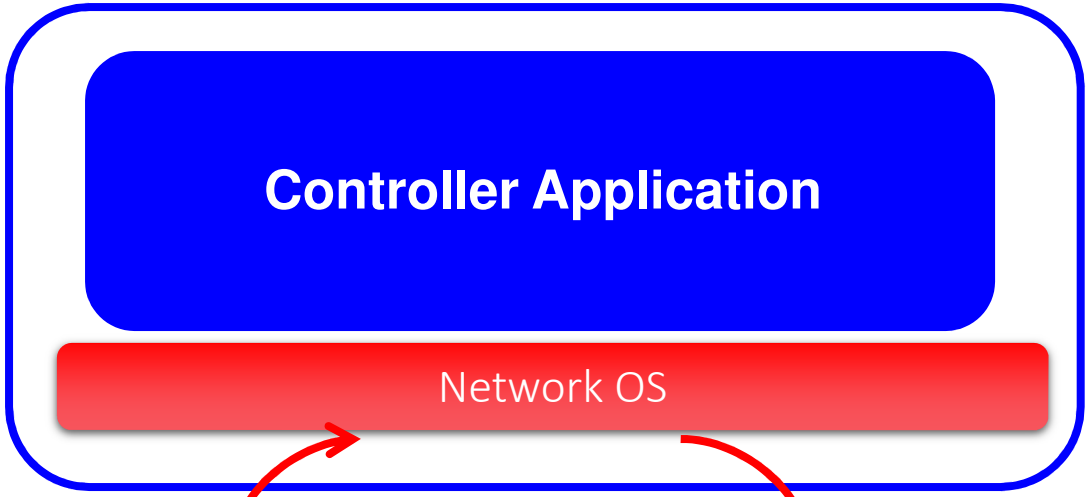
- Simple packet-handling rules
  - Pattern: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

Unifies Different Kinds of Boxes

- Router
  - Match: longest destination IP prefix
  - Action: forward out a link
- Switch
  - Match: destination MAC address
  - Action: forward or flood
- Firewall
  - Match: IP addresses and TCP/UDP port numbers
  - Action: permit or deny
- NAT
  - Match: IP address and port
  - Action: rewrite address and port

Controller: Programmability



Events from switches
Topology changes,
Traffic statistics,
Arriving packets

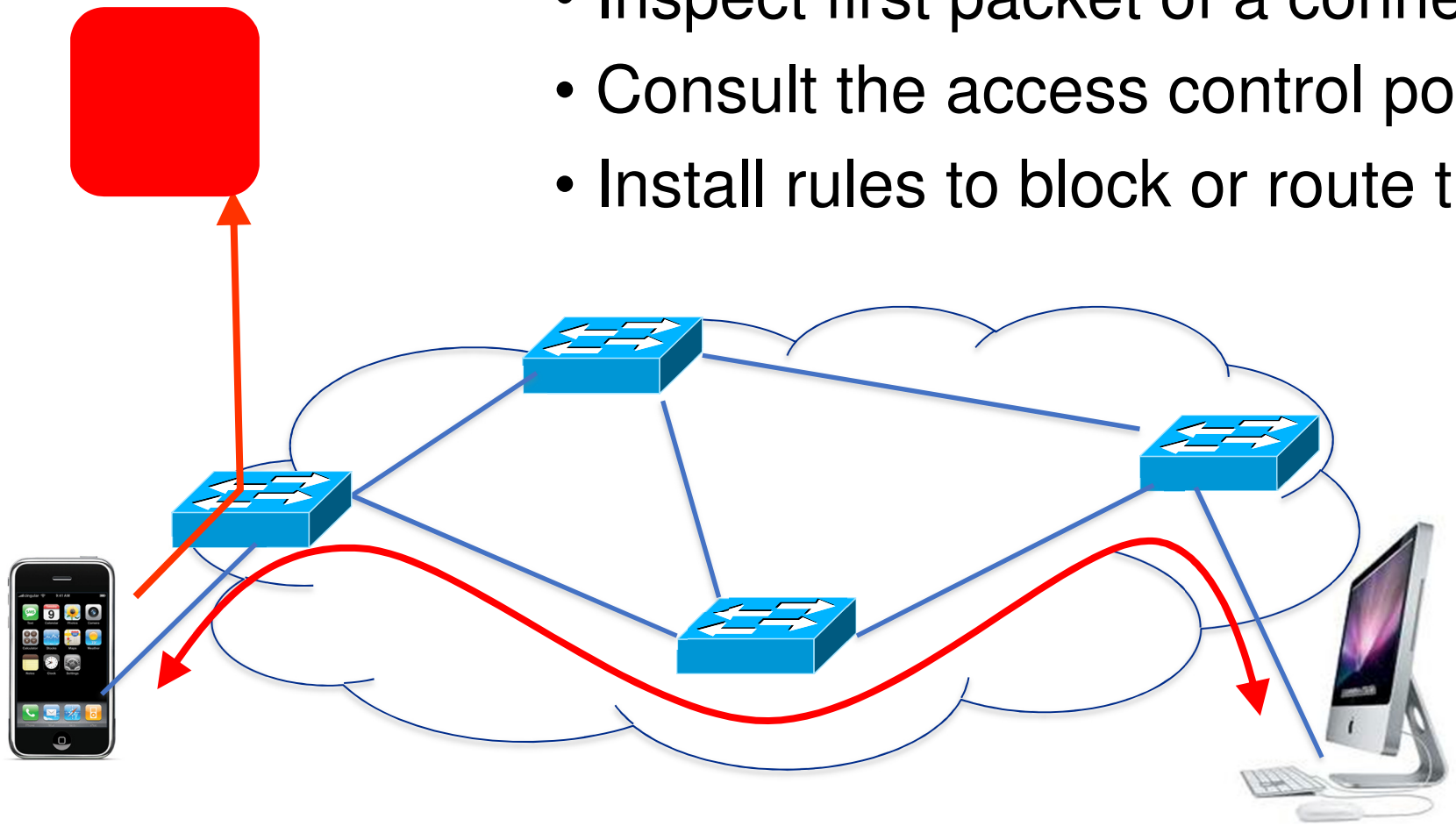
Commands to switches
(Un)install rules,
Query statistics,
Send packets

Example OpenFlow Applications

- Dynamic access control
- Seamless mobility/migration
- Server load balancing
- Network virtualization
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection

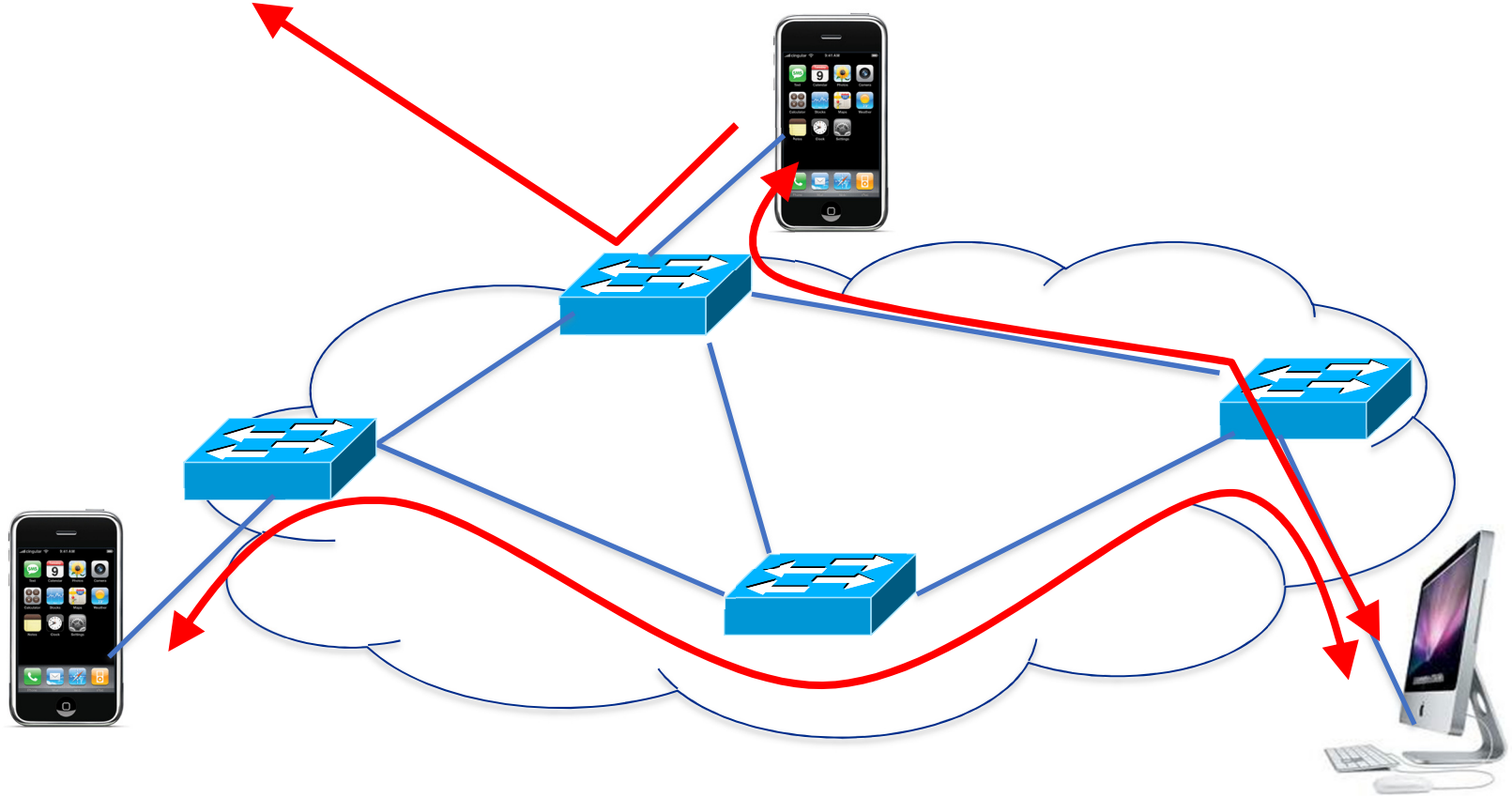
Example: Dynamic Access Control

- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic



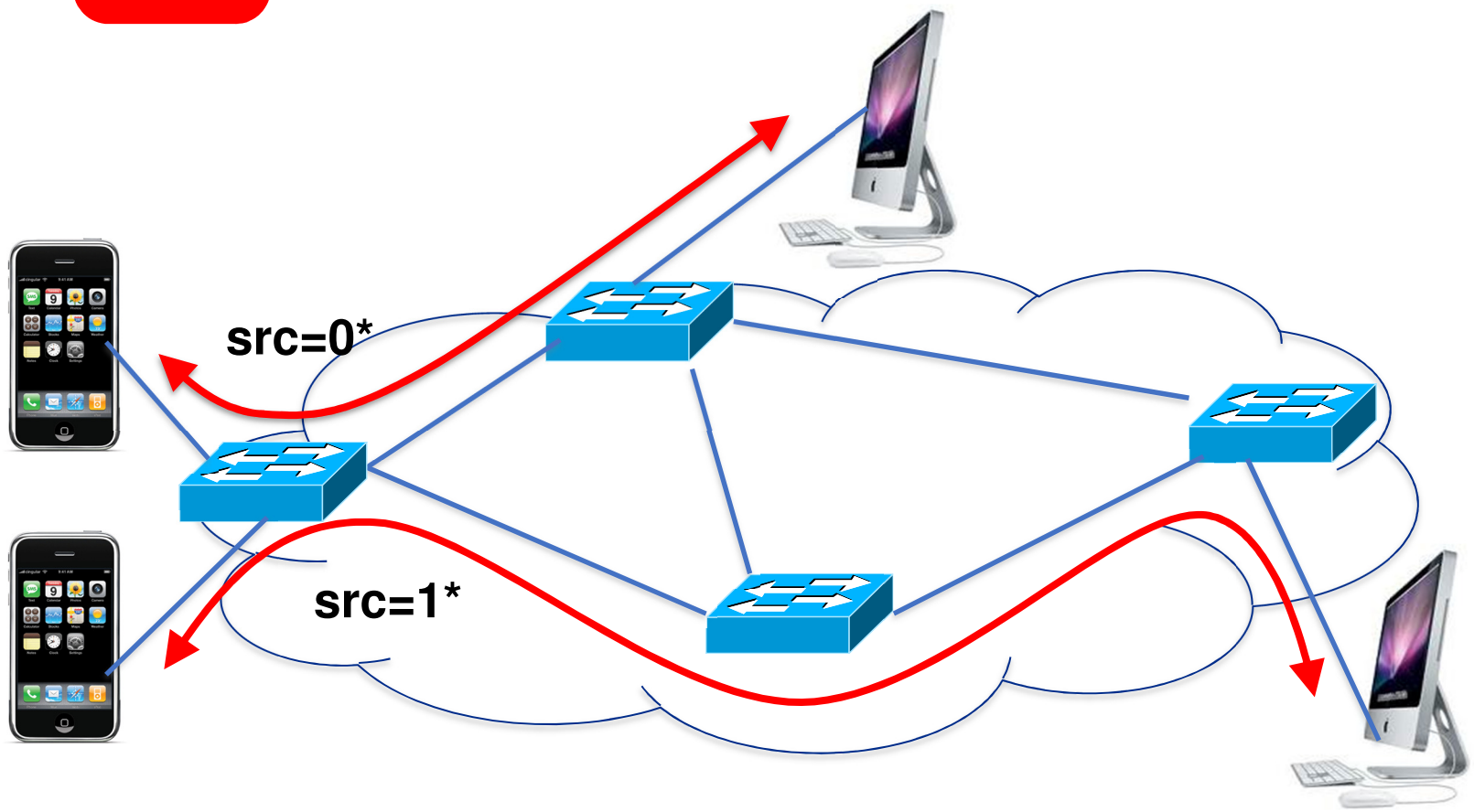
Seamless Mobility/Migration

- See host send traffic at new location
- Modify rules to reroute the traffic



Server Load Balancing

- Pre-install load-balancing policy
- Split traffic based on source IP



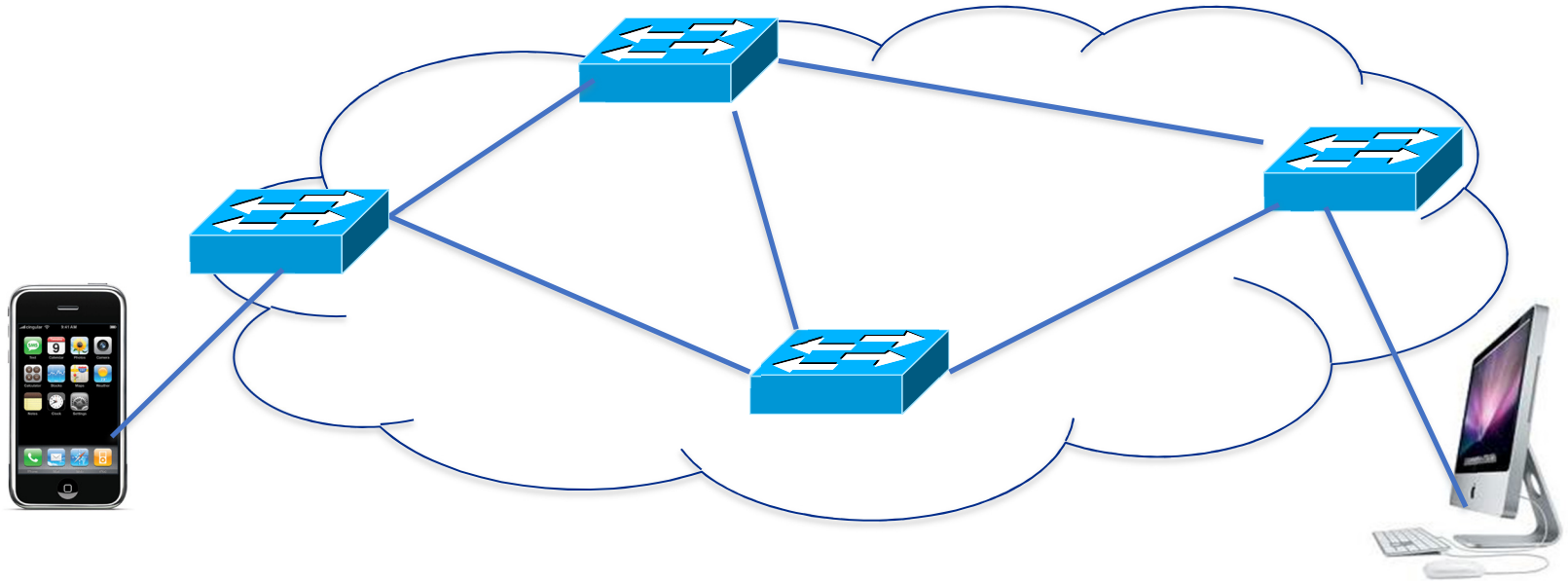
Network Virtualization

Controller #1

Controller #2

Controller #3

Partition the space of packet headers

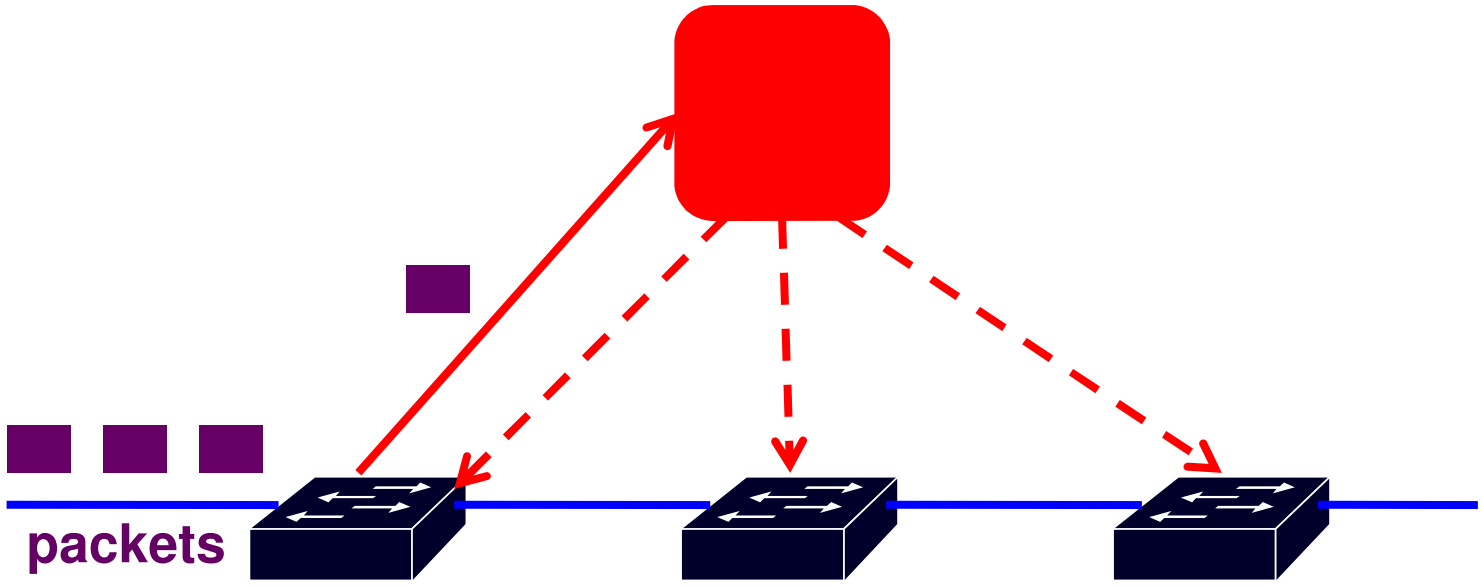


OpenFlow in the Wild

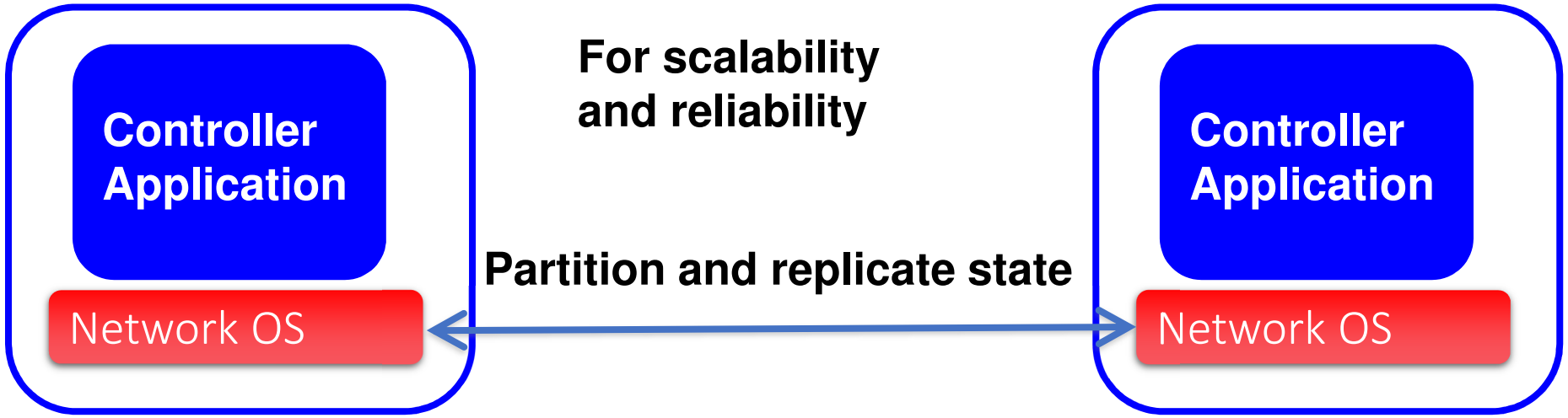
- Open Networking Foundation
  - Google, Facebook, Microsoft, Yahoo, Verizon, Deutsche Telekom, and many other companies
- Commercial OpenFlow switches
  - HP, NEC, Quanta, Dell, IBM, Juniper, …
- Network operating systems
  - NOX, Beacon, Floodlight, Nettle, ONIX, POX, Frenetic
- Network deployments
  - Eight campuses, and two research backbone networks
  - Commercial deployments (e.g. Google backbone)

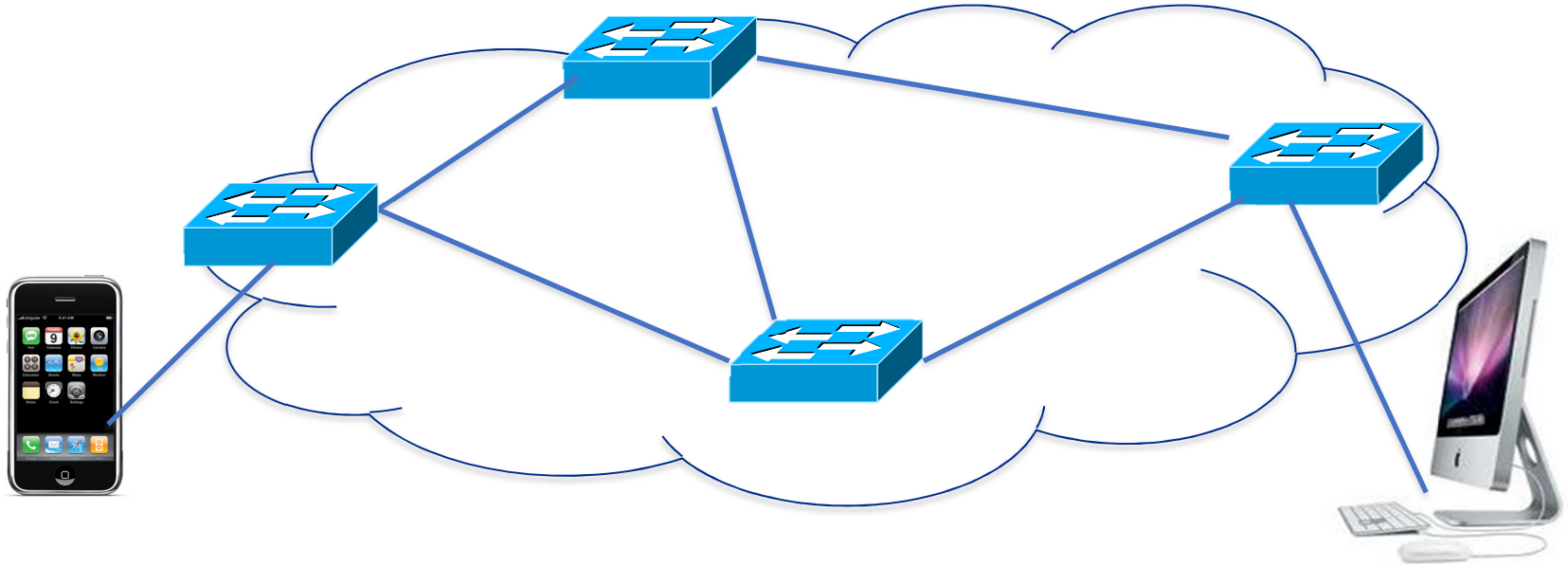
Controller Delay and Overhead

- Controller is much slower than the switch
- Processing packets leads to delay and overhead
- Need to keep most packets in the “fast path”



Distributed Controller

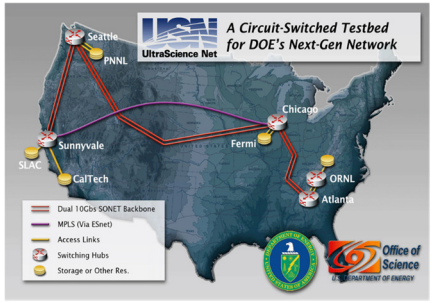


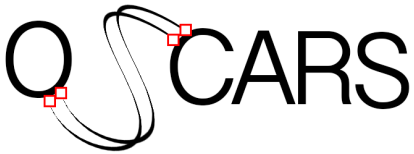


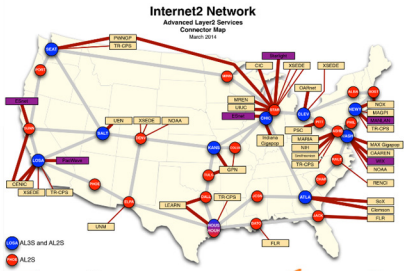
High-performance Networks

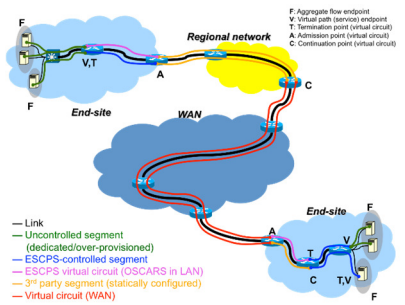
- HPN testbeds and projects

Projects	Remarks
USN	DOE/DOD ultra-science networks
OSCARS	Bandwidth reservation within the ESnet
AL2S (ION)	Bandwidth reservation within the Internet2
DYNES	Edge network bandwidth reservation for the Internet2
DRAGON	Using MPLS technology
CHEETAH	Circuit-switched high-speed end-to-end transport arch testbed
TeraPaths	End-to-end virtual path with bandwidth guarantees
ESCPS	Provisioning end-to-end inter-domain dynamics circuits
UCLP	Network resources treated as software objects
JPN(2/2plus)	Fully-fledged next-generation testbed for research
Geant2	Funded by NERNs and EC, testbed for research in European
HOPI	Combining packet- and circuit-switching
GENI	Virtual laboratory for exploring future Internet
B4	Google's globally-deployed software defined WAN



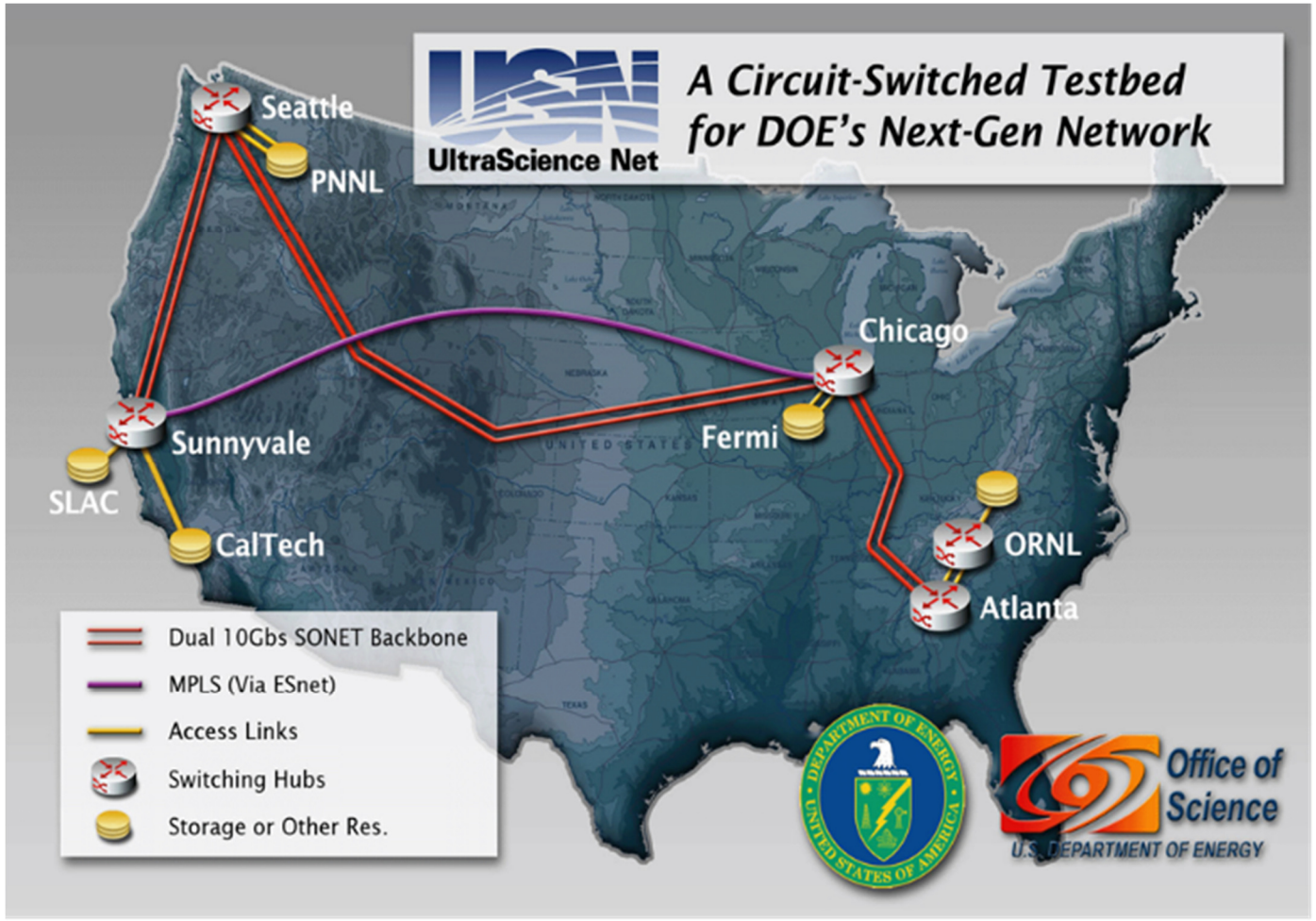






UltraScience Net – In a Nutshell

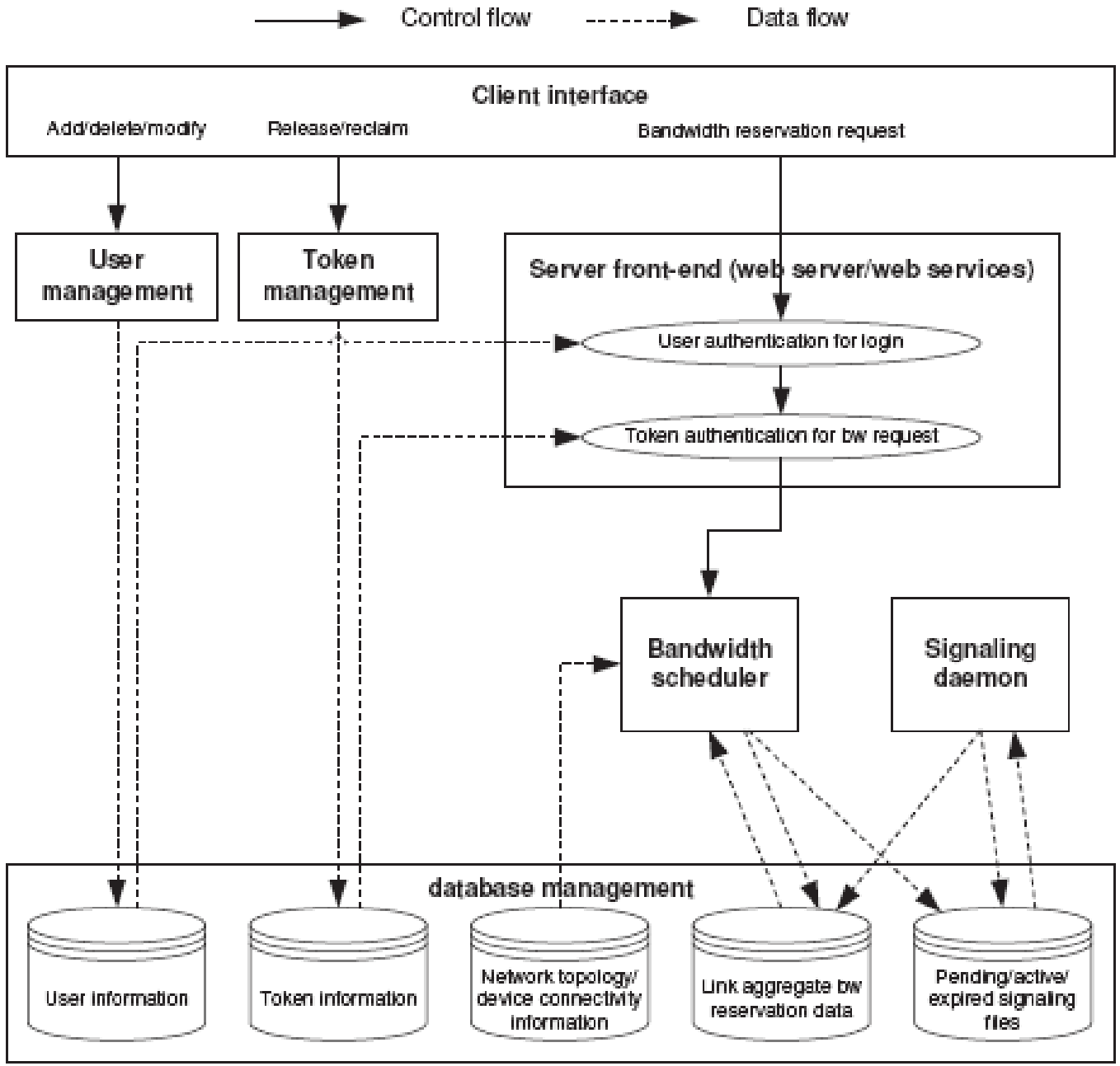
- Experimental Network Research Testbed



https://www.csm.ornl.gov/ultranet/

Control Plane

- Responsible for
  1. Reserving link bandwidths
  2. Setting up end-to-end network paths
  3. Releasing resources when tasks are completed

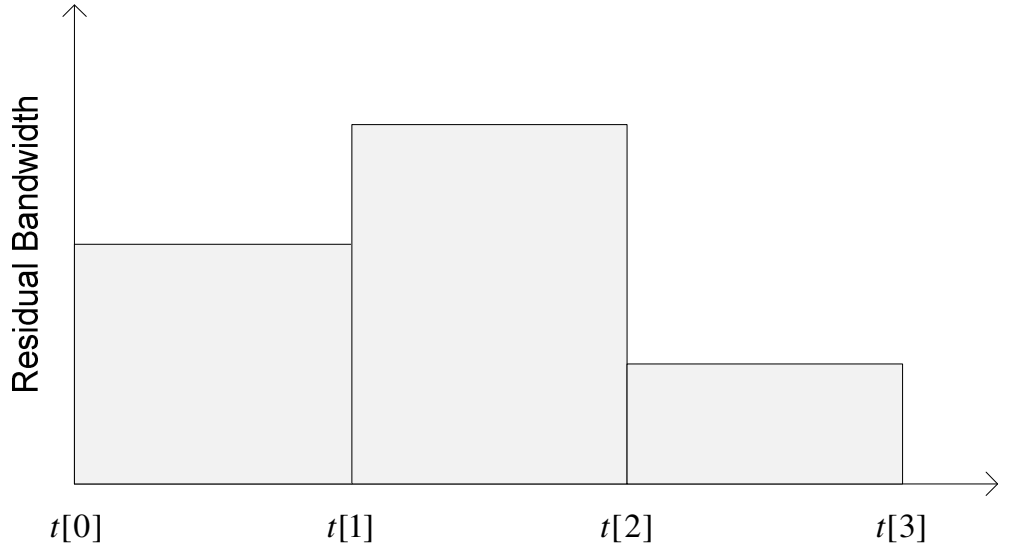


Bandwidth Scheduler

- Central component
  - Computes paths and allocate link bandwidths
- Scheduling in USN
  - Fixed slot: start time, end time, target bandwidth
    - Extensions of Dijkstra's Algorithm
  - All slots: duration, target bandwidth
    - Extensions of Bellman-Ford Algorithm
  - Both are solvable by polynomial-time algorithms

Network Model for Bandwidth Scheduling

- Graph: G(V, E)
- Link bandwidth
  - Segmented constant functions of time
  - Time-bandwidth (TB): (t[i], t[i+1], b[i])



- Aggregated TB for all links

4 Types of Scheduling Problems

- Given: $G = (E, V)$, ATB, v_s , v_d , data size δ
- Objective: minimize data transfer end time
- Fixed Path with Fixed Bandwidth (FPFB)
  - Compute a fixed path from v_s to v_d with a constant (fixed) bandwidth
- Fixed Path with Variable Bandwidth (FPVB)
  - Compute a fixed path from v_s to v_d with varying bandwidths across multiple time slots
- Variable Path with Fixed Bandwidth (VPFB)
  - Compute a set of paths from v_s to v_d at different time slots with the same (fixed) bandwidth
- Variable Path with Variable Bandwidth (VPVB)
  - Compute a set of paths from v_s to v_d at different time slots with varying bandwidths across multiple time slots

Lin and Wu. Complexity Analysis and Algorithm Design for Advance Bandwidth Scheduling in Dedicated Networks. ACM/IEEE Transactions on Networking, 2013.

VPFB & VPVB

- Multiple paths are used in a sequential order

- Path switching incurs a delay (overhead) τ
  - Path switching delay is negligible ($\tau = 0$)
    - VPFB-0, VPVB-0
  - Path switching delay is not negligible ($\tau > 0$)
    - VPFB-1, VPVB-1

- When τ is large enough
  - VPFB-1 → FPFB, VPVB-1 → FPVB

Problem Features

- FPFB is the most stringent
- VPVB is the most flexible
- FPFB and VPFB restrict the bandwidth
  - Not always optimal to start data transfer immediately
  - Suited for transport methods with fixed rate
    - FRTP, PLUT, Tsunami, Hurricane, RBUDP
- FPVB and VPVB use variable bandwidth
  - Always start immediately
  - Suited for transport methods with dynamically adapted rate
    - SABUL/UDT, RAPID, RUNAT, improved TCP

Complexity and Algorithm

Problem	Complexity	Algorithm
FPFB	P	$OptFVFB$
FPVB	NP-complete	$OptFPVB$, $MinFPVB$
VPFB-0	P	$OptVPFB-0$
VPFB-1	P	$OptVPFB-1$
VPVB-0	P	$OptVPVB-0$
VPVB-1	NP-complete	$OptVPVB-1$, $MinVPVB-1$

- An optimal algorithm for each problem
- A heuristic for each NPC problem

Lin and Wu. Complexity Analysis and Algorithm Design for Advance Bandwidth Scheduling in Dedicated Networks. ACM/IEEE Transactions on Networking, 2013.

Big Data Transfer

Goals

- Composition of an end-to-end data transfer path
  - Resources/services discovery, and selection
  - Dedicated connection with (high) bandwidth reserved

- Selection and configuration of transport methods
  - Selection of appropriate transport methods
  - Specification of control parameters results in good performance

Transport-Support Workflow Composition and Optimization

- Objective: to choose a subset of modules across all *K* zones (layers) to compose an end-to-end data transfer path to meet the user request *r* with the maximum profit

$$\max \sum_{j=1}^{K} \sum_{i=1}^{N_j} P_{i,j} \cdot x_{i,j} = \max \sum_{j=1}^{K} \sum_{i=1}^{N_j} \sum_{k=1}^{n} \alpha_k \cdot p_{i,j,k} \cdot x_{i,j}$$

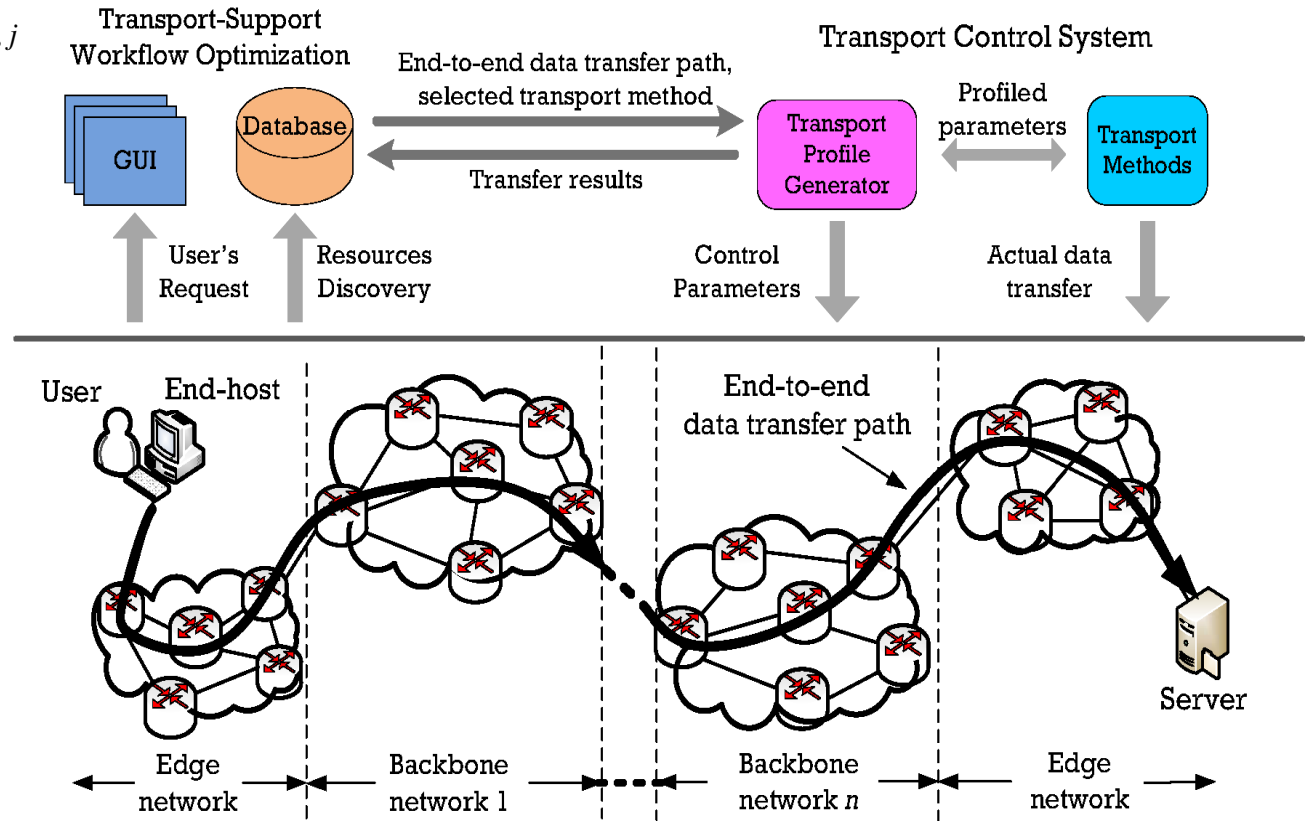
subject to

$$\sum_{j=1}^{K} \sum_{i=1}^{N_j} C_{i,j} \cdot x_{i,j} \le C_{\max}$$

$$\sum_{i=1}^{N_j} x_{i,j} = 1,\ j = 1, 2, \ldots, K$$

$$x_{i,j} \in \{0,1\},\ j = 1, 2, \ldots, K;\ i = 1, 2, \ldots, N_j$$

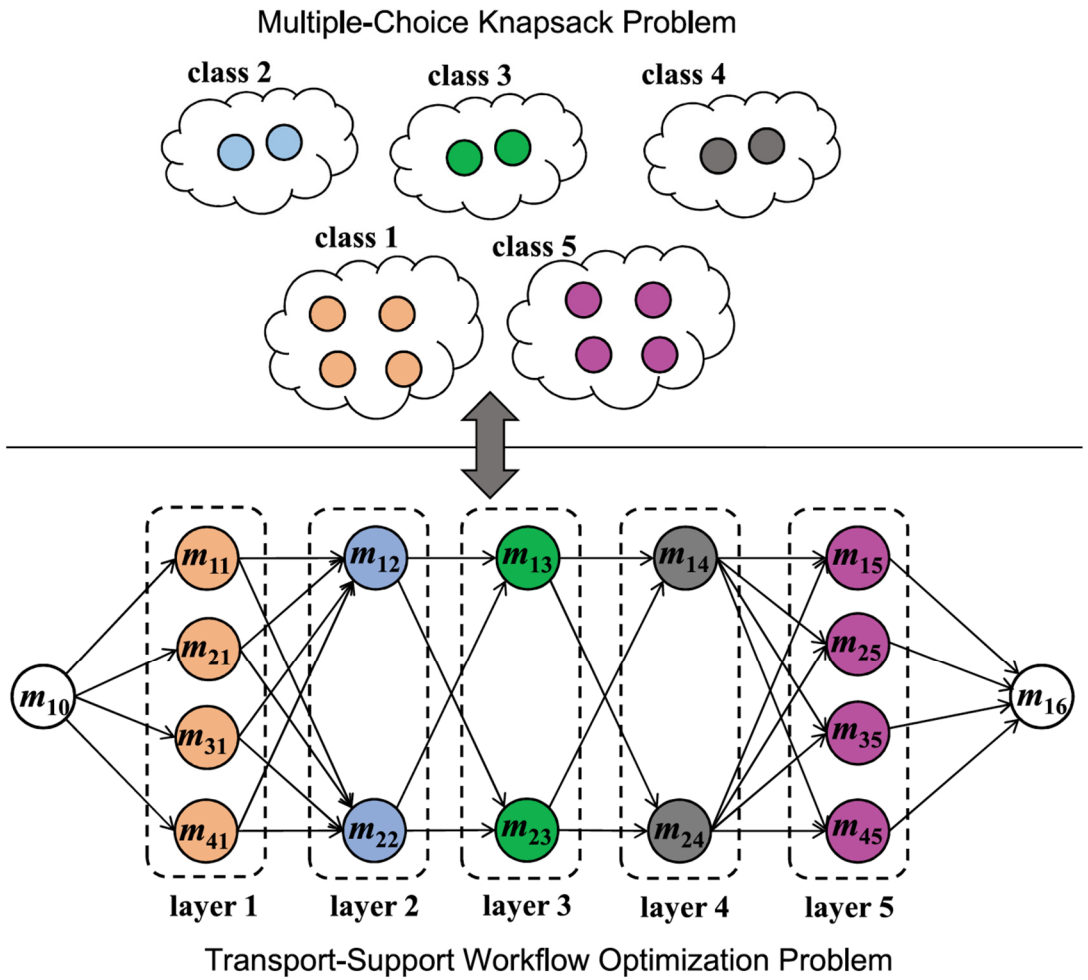
$$\prod_{j=1}^{K-1} e_{(i_j^*, j),(i_{j+1}^*, j+1)} \cdot x_{i_j^*, j} \cdot x_{i_{j+1}^*, j+1} = 1$$



Yun et al. Transport-Support Workflow Composition and Optimization for Big Data Movement in High-Performance Networks. IEEE TPDS, 2017.

Complexity and Algorithms

- When the cost limit is infinite
  - If we are free to use the advanced services, or
  - If we do not care the cost
  - The problem becomes longest profit path problem in a Directed Acyclic Graph (DAG)
- When the cost limit is finite
  - Weakly NP-complete problem
  - Proof: reduce from Multiple-Choice Knapsack Problem (MCKP)



Yun et al. Transport-Support Workflow Composition and Optimization for Big Data Movement in High-Performance Networks. IEEE TPDS, 2017.

Complexity and Algorithms

- When the cost limit is infinite
  - Critical Path-based Module Selection (CPMS)
  - Linear time
- When the cost limit is finite
  - Dynamic Programming-based Module Selection (DPMS)
  - It is optimal, but runs in pseudo-polynomial time
  - Optimal sub-structure

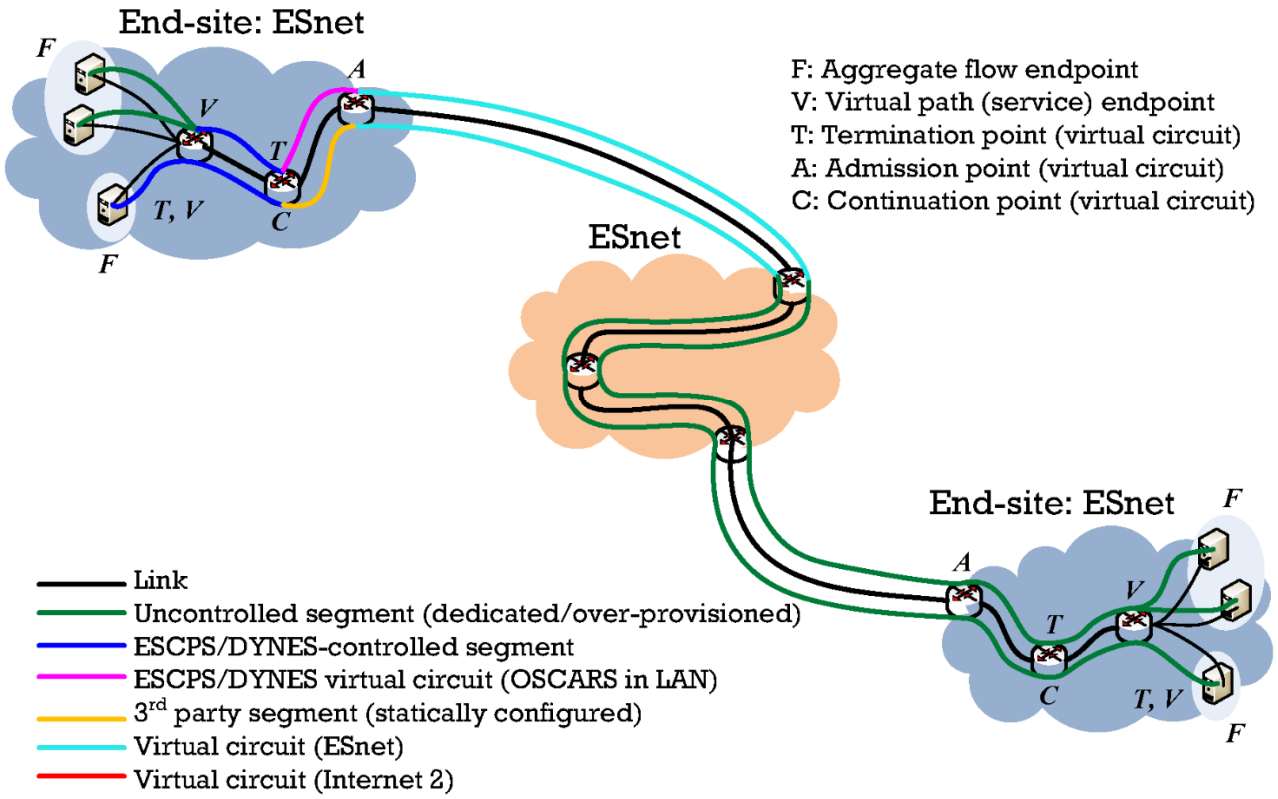
$$D(C,i,j) = P_{i,j} + \max\left\{ D\left(C - C_{i,j}, i', j-1\right) \middle| \begin{matrix} e_{(i',j-1),(i,j)} = 1, \\ \left(C - C_{i,j}, i', j-1\right) \geq 0 \end{matrix} \right\}$$

D(C, i, j) is the maximal achievable profit using modules from zone 0 to zone j when module $m_{i,j}$ is selected under the cost limit C

Yun et al. Transport-Support Workflow Composition and Optimization for Big Data Movement in High-Performance Networks. IEEE TPDS, 2017.

Experiment Results

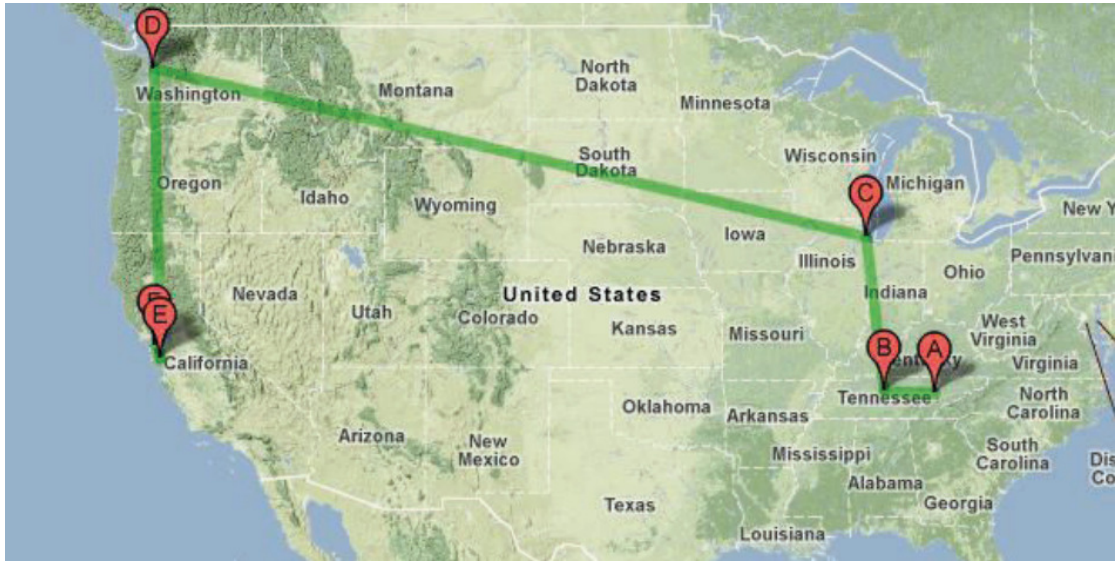
- Case 1: data transfer from ORNL to LBNL
- Layout of network segments



Yun et al. Transport-Support Workflow Composition and Optimization for Big Data Movement in High-Performance Networks. IEEE TPDS, 2017.

Experiment Results

- Case 1: data transfer from ORNL to LBNL
- GridFTP → ESCPS → OSCARS → GridFTP



Source (A): ornl.gov **Destination (F):** lbl.gov

Network: OSCARS **Max. Backbone Capacity:** 5536 Mb/s

Available Bandwidth:

A→B: 10000 Mb/s (10GE-Link)

B→C: 8897 Mb/s **C→D:** 5536 Mb/s

D→E: 9067 Mb/s **E→F:** 10000 Mb/s (2*MAN 10G RING)

Path:

A: ORNL (ornl.gov) → **ORNL** (site)

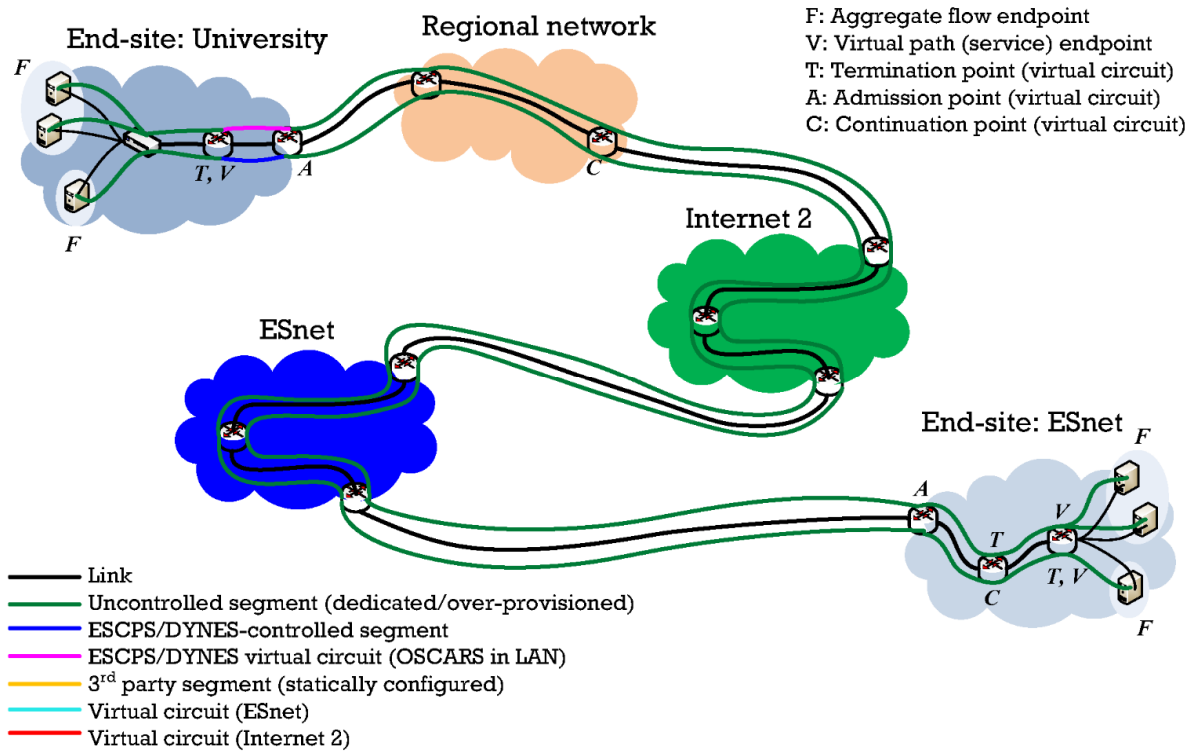
B: (NASH) → **NASH** **C:** (STAR) → **STAR**

D: (PNWG) → **PNWG** **E:** (SUNN) → **SUNN**

F: LBNL (lbl.gov) → **LBNL**

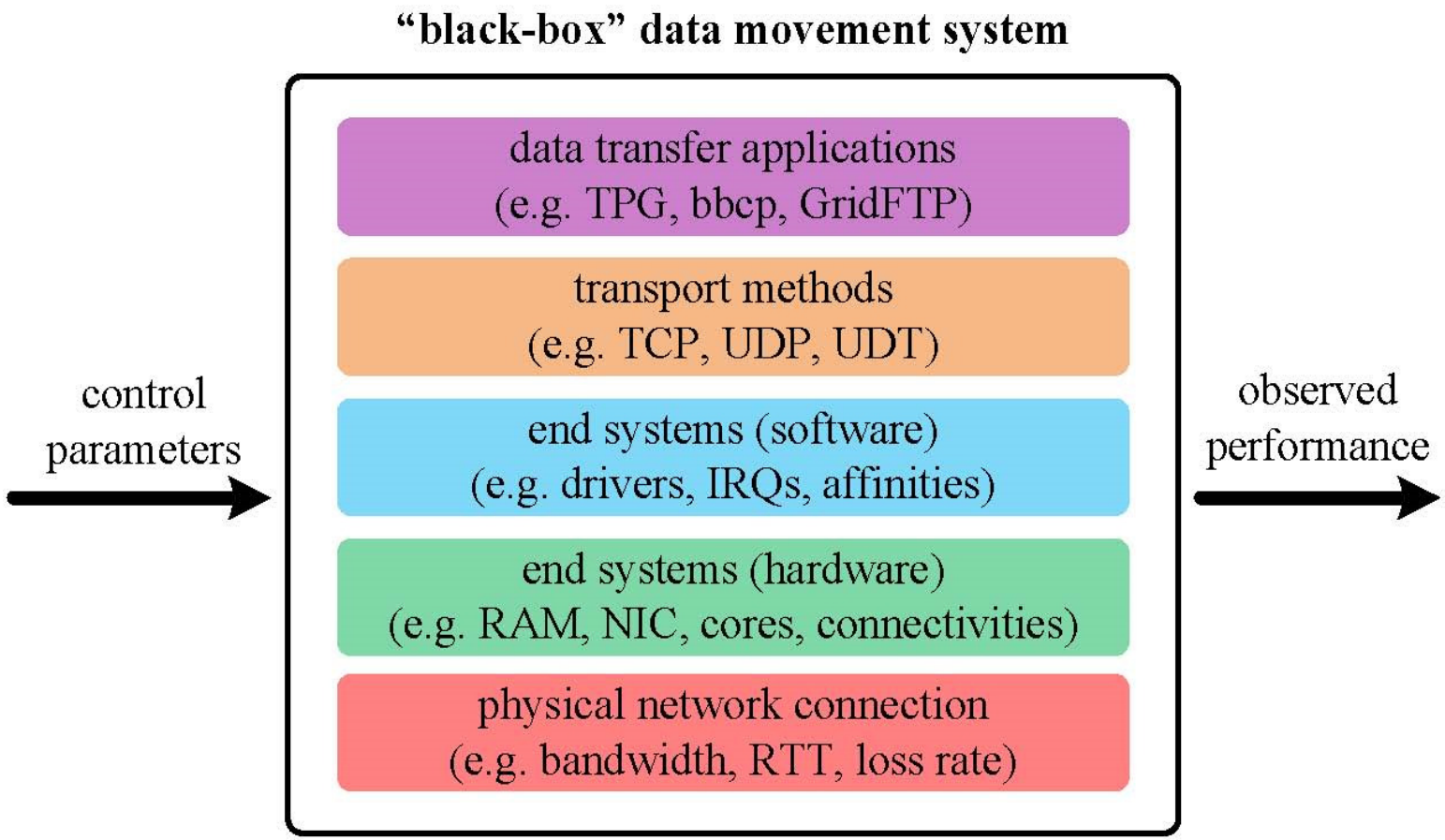
Experiment Results

- Case 2: data transfer from LLNL to UChicago
- GridFTP → ESCPS → OSCARS → ION → DYNES → GridFTP



Yun et al. Transport-Support Workflow Composition and Optimization for Big Data Movement in High-Performance Networks. IEEE TPDS, 2017.

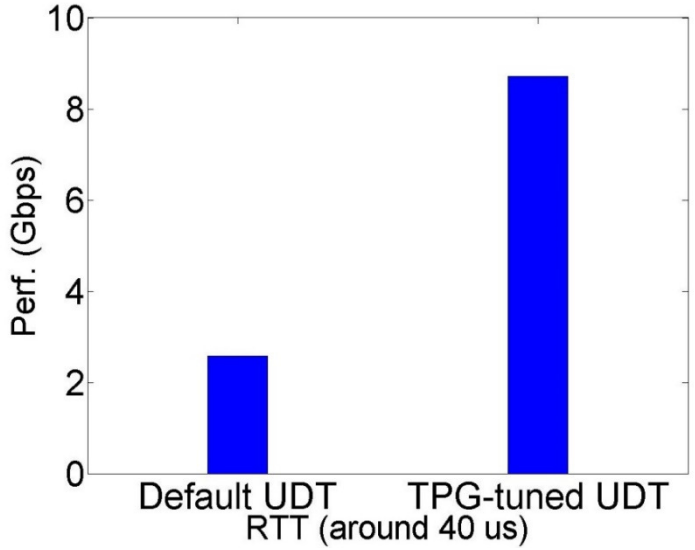
Big Data Movement



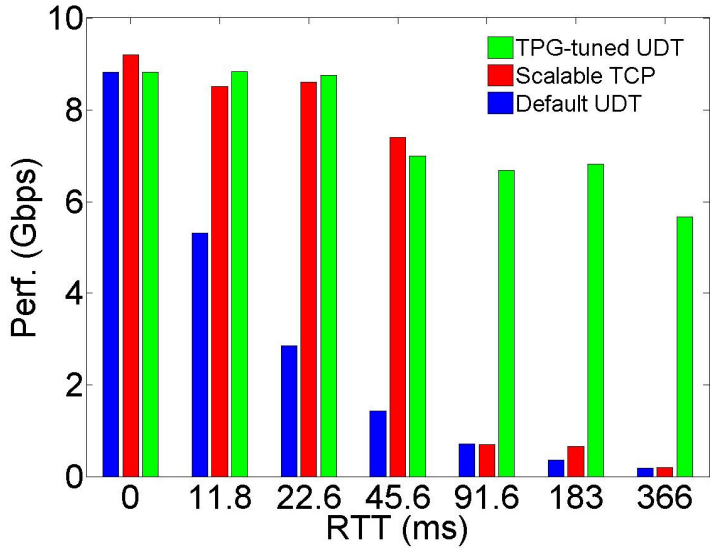
Transport Profiling

- What can it do?

NJIT local testbed (0.04ms)



ORNL emulated testbed (0-366ms)

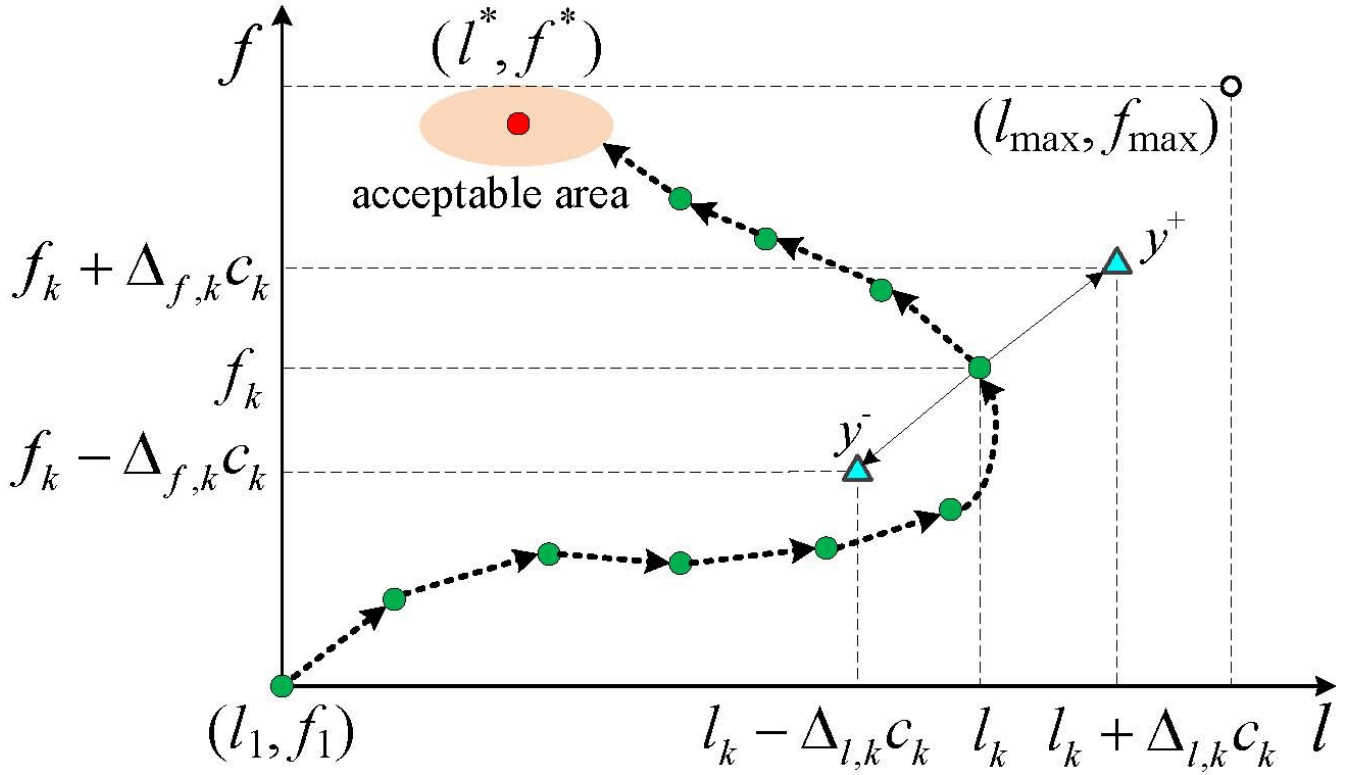


Transport Profiling

- Profiling can help improve transport performance
- Exhaustive approach is prohibitively time-consuming
- UDT example
  - Block size: 1 to 25 times of payloads
  - Buffer size: 1MB to 1GB with a 2MB resolution
  - Each “one-time” profiling takes 2 minutes
  - Total: 2 minutes x (25 x (1024 / 2 + 1)) = 25,650 minutes
    ≈ 18 days → impractical in reality

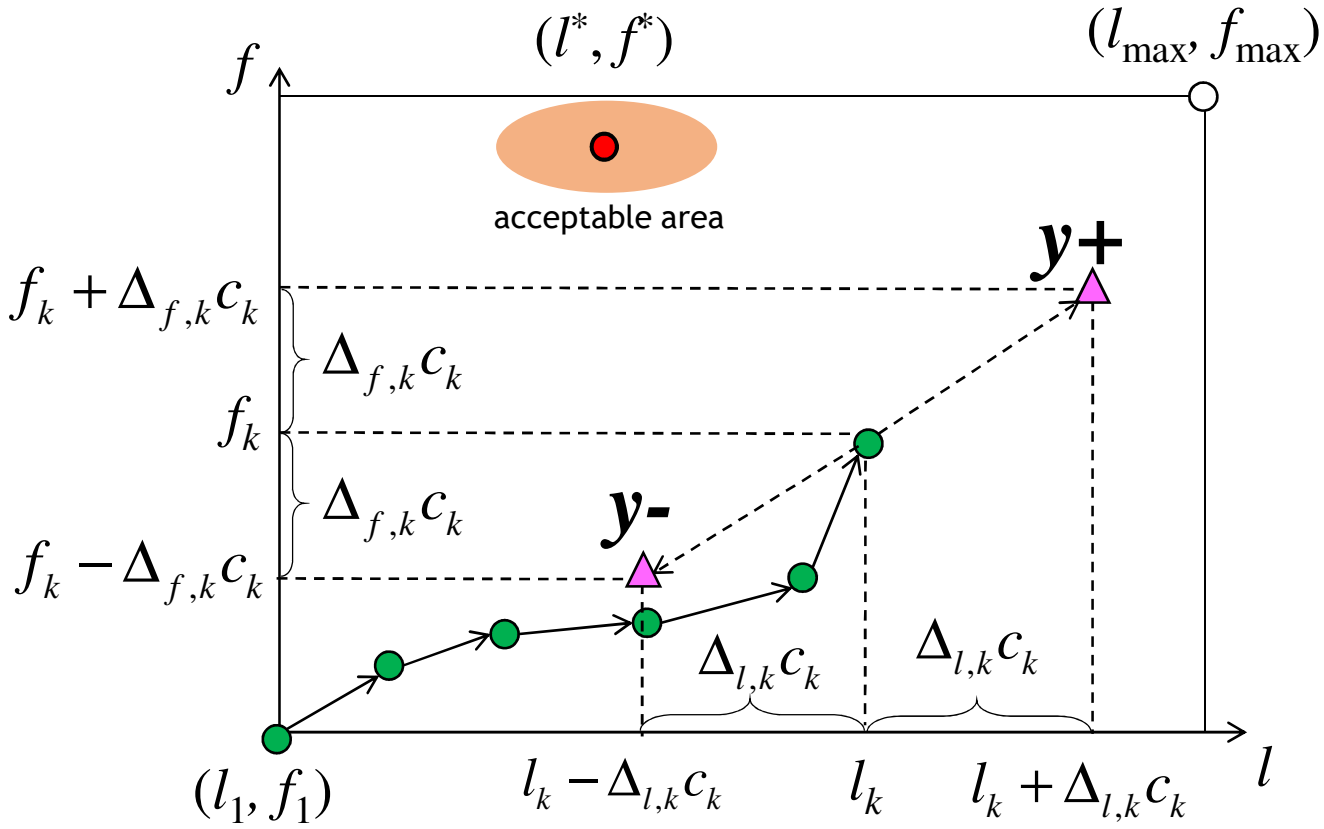
Profiling based on Stochastic Approximation

- Visualization of the general profiling process



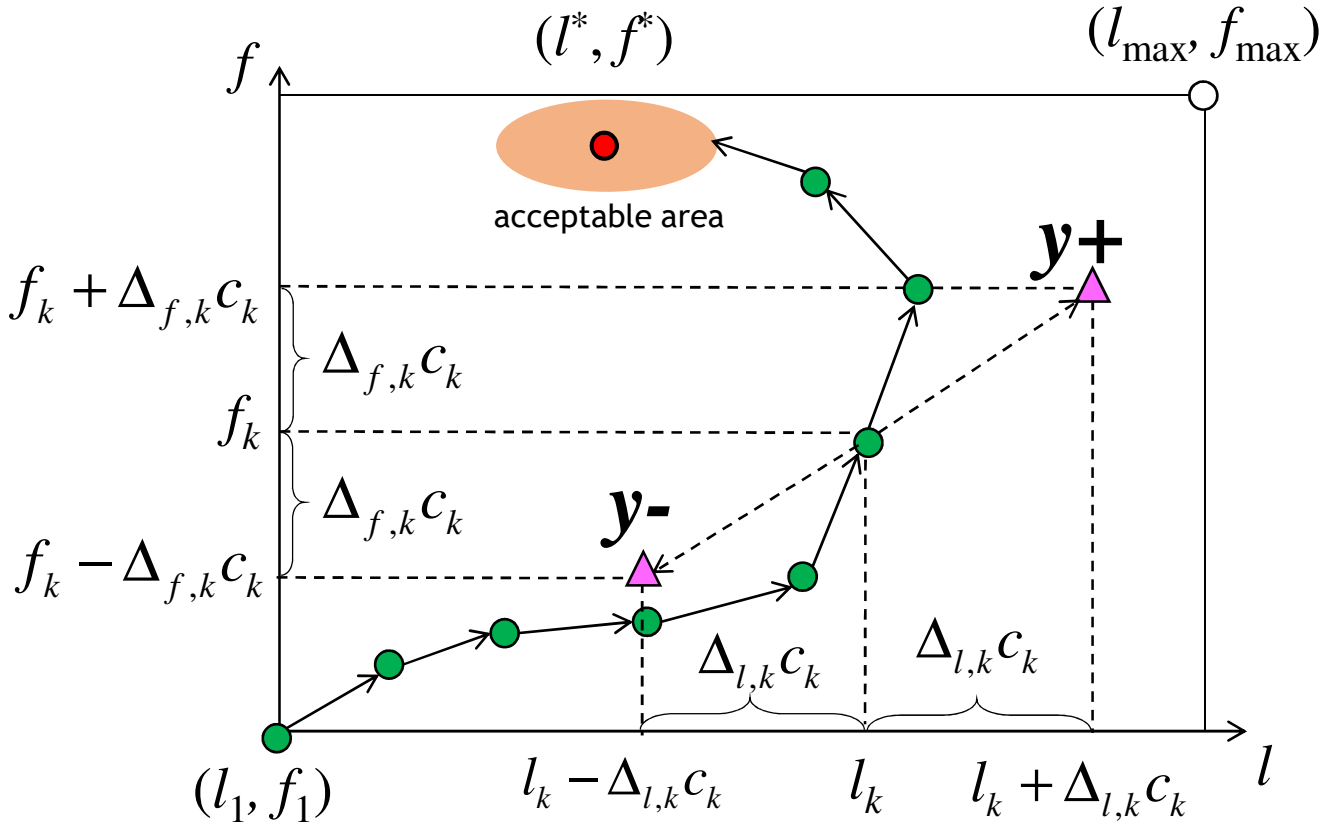
Profiling based on Stochastic Approximation

- Perform two profiling tests with control parameter values calculated based on the perturbations



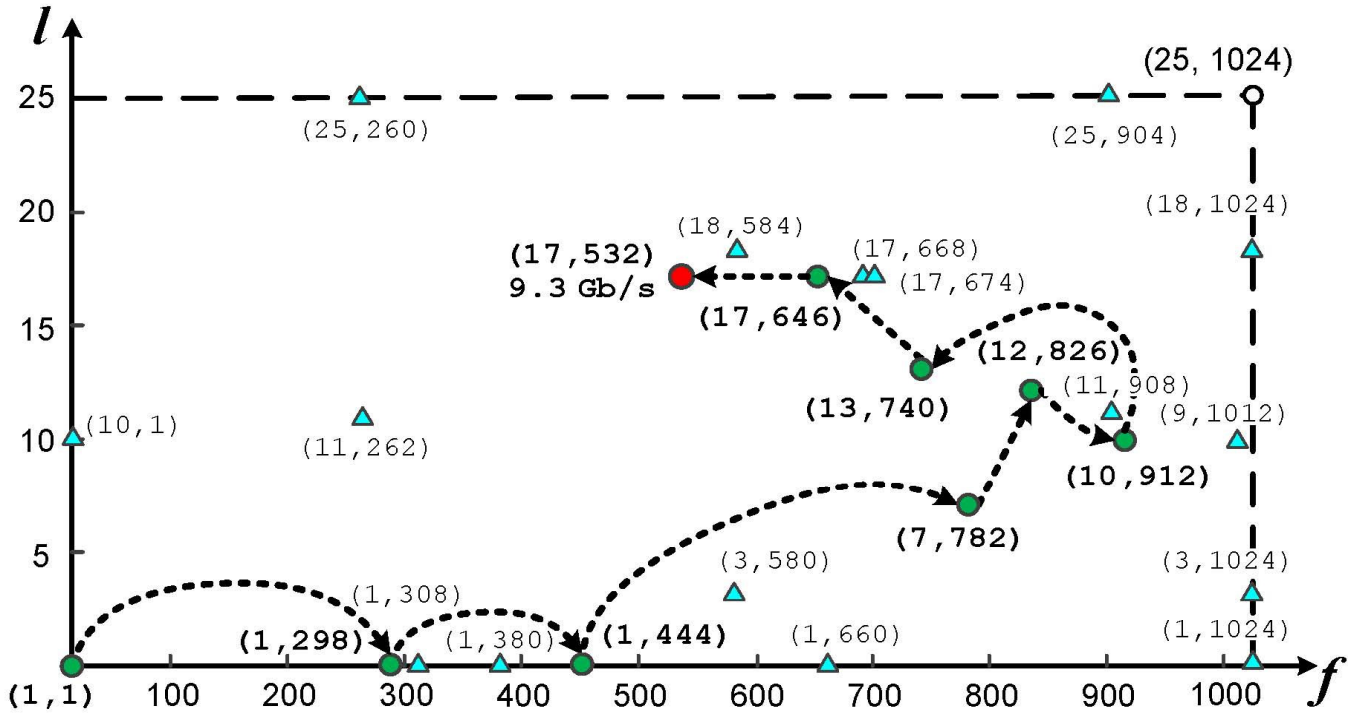
Profiling based on Stochastic Approximation

- Continue such iterations until termination conditions are met or the performance reaches the acceptable area



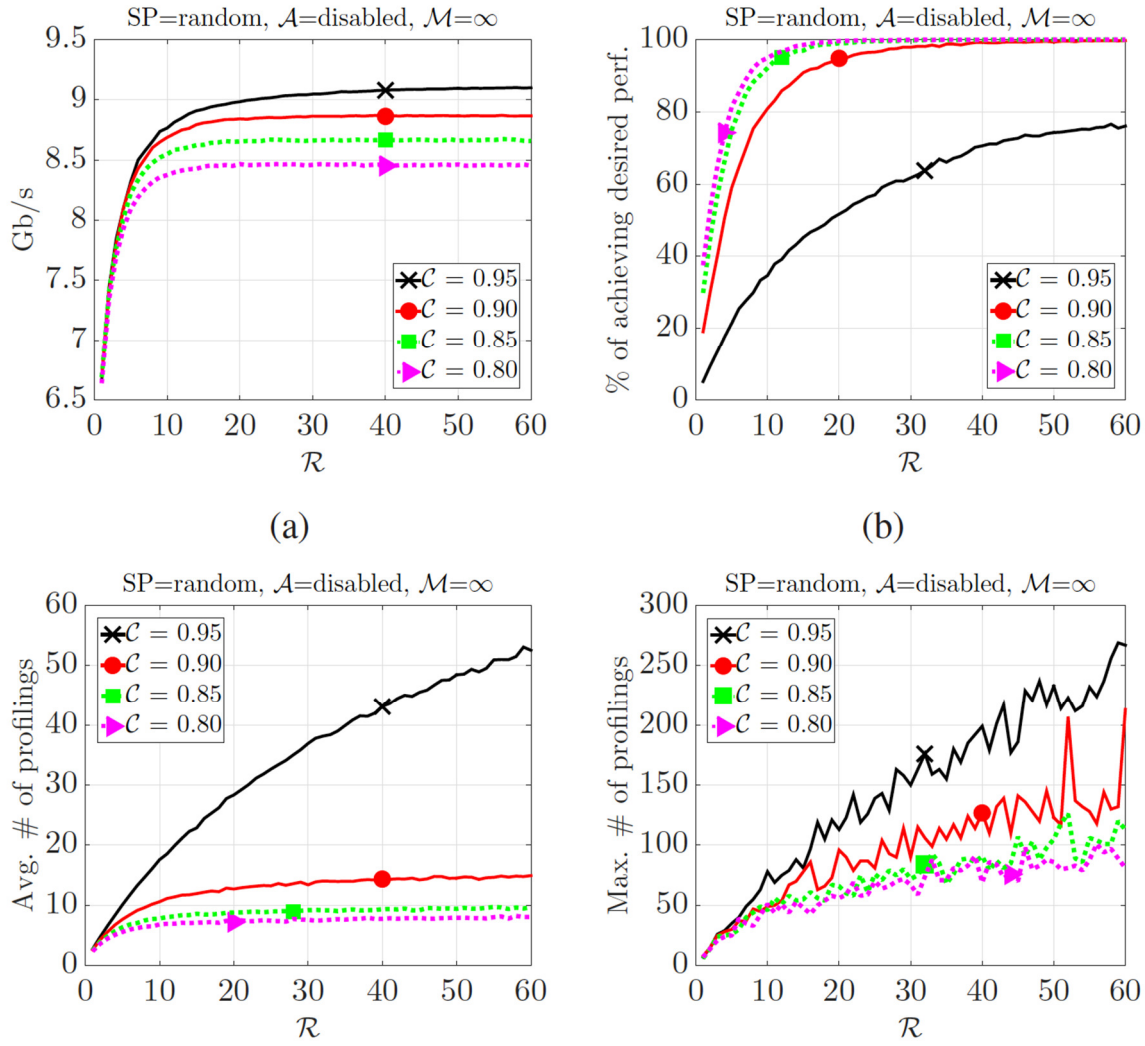
Profiling based on Stochastic Approximation

- Profiling process trace



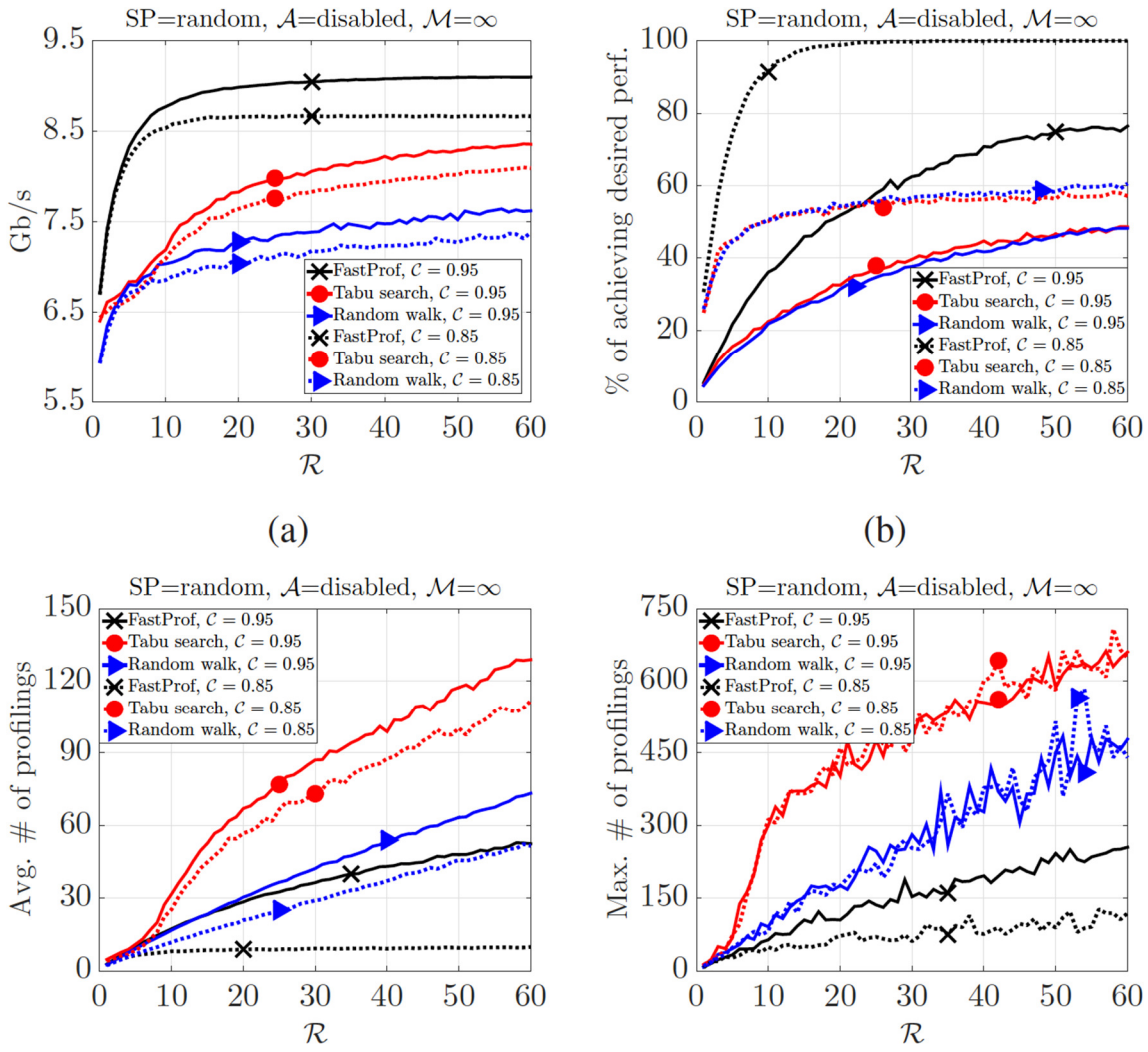
Profiling based on Stochastic Approximation

- Overall performance
- R: allowed number of iterations without performance improvement
- Achieved throughput performance
- % of achieving desired performance
- Average profiling time (avg. case)
- Maximal profiling time (worse case)



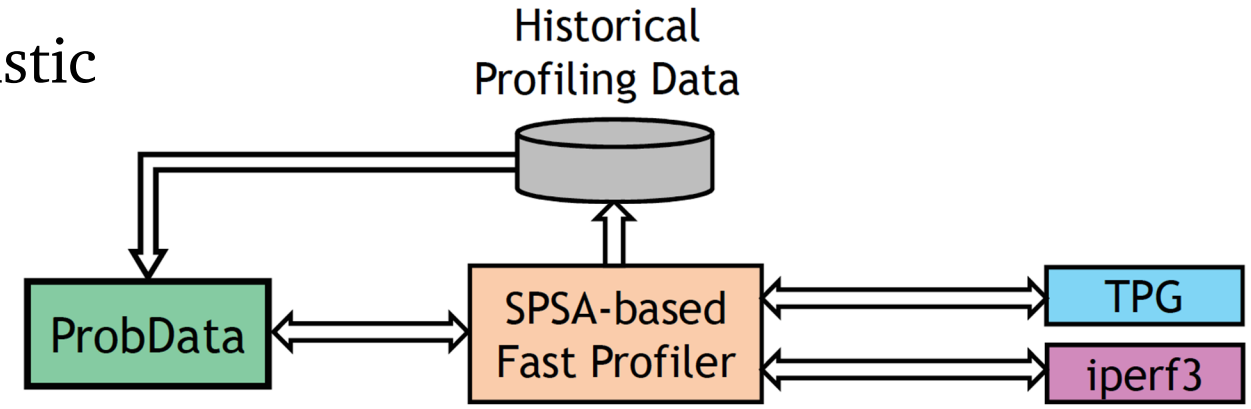
Profiling based on Stochastic Approximation

- Comparison with *Tabu search* and *random walk*



PRofiling Optimization Based DAta Transfer Advisor (ProbData)

- Method: profiling based on stochastic approximation
- TCP profiler: ESnet iperf3
- UDT profiler: TPG
- Storage of historical profiling data

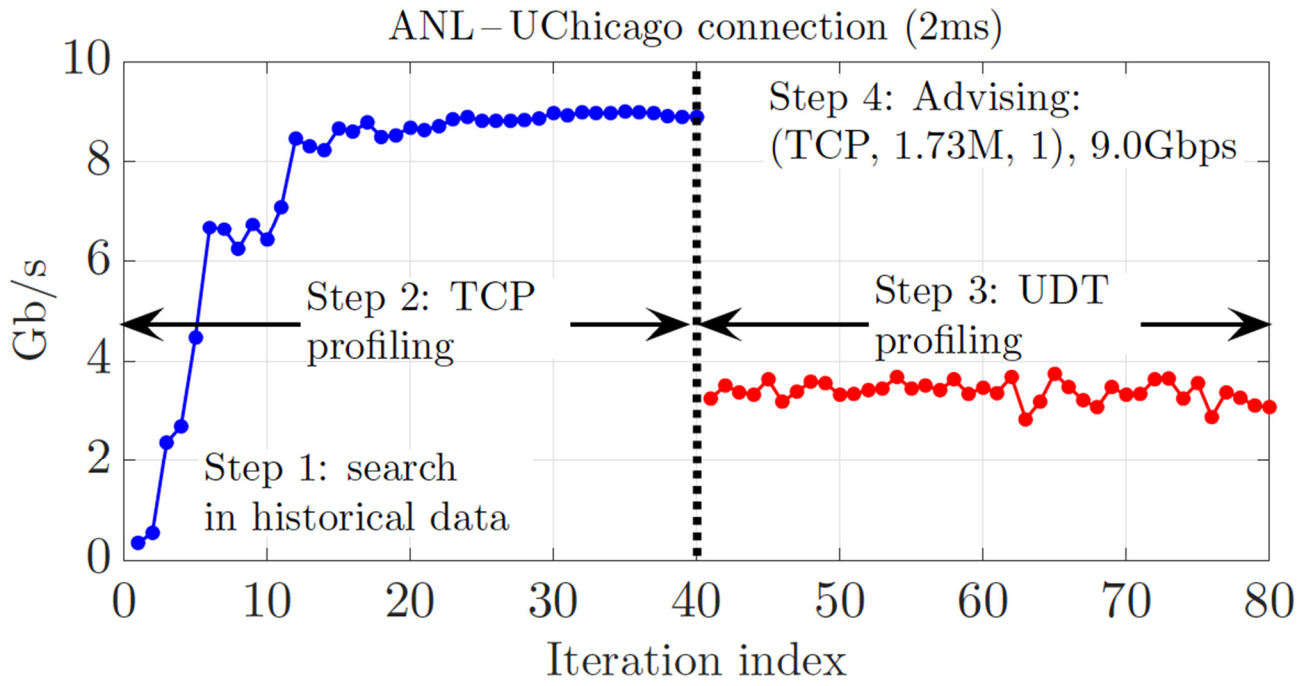


ProbData stores the intermediate profiling results in each SPSA-based profiling iteration for future use

Yun et al. Data Transfer Advisor with Transport Profiling Optimization. In Proc. of LCN 2017, 2017.

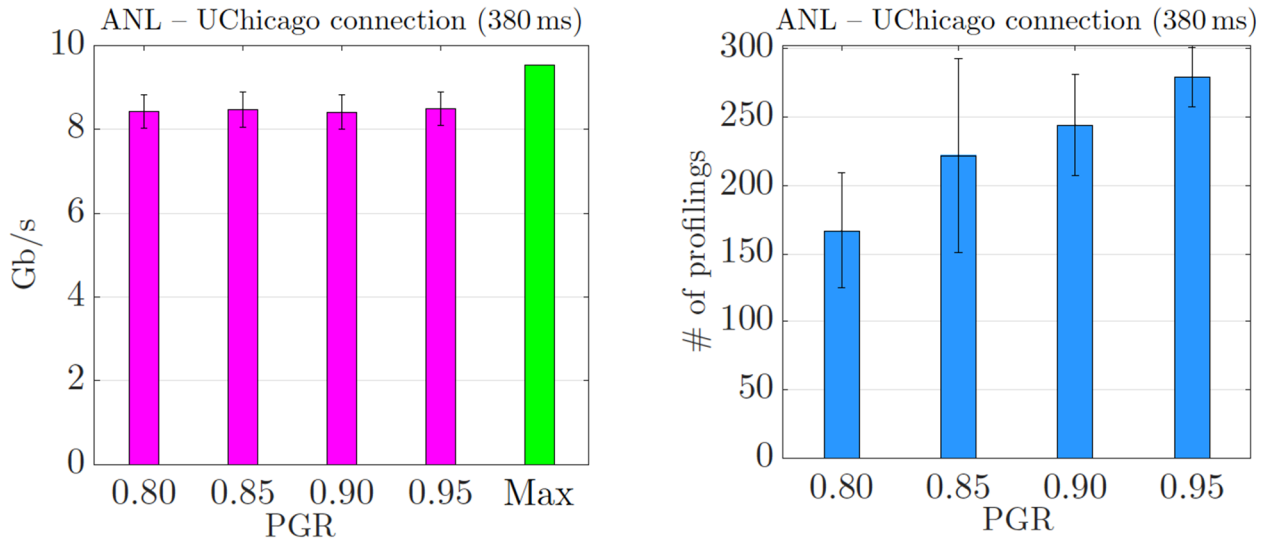
Experimental Results

- Advising procedure
- ANL→UChicago 10Gbps 2ms physical connection

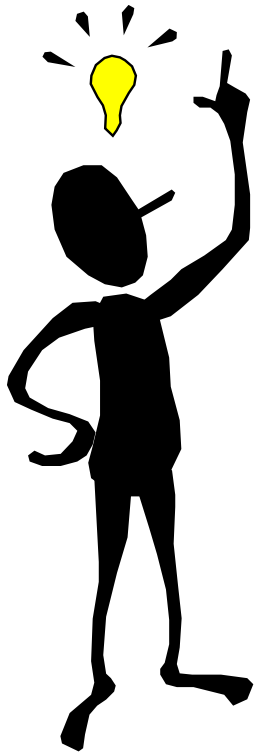


Experimental Results

- Experimental evaluation
- ANL→UChicago 10Gbps 380ms connection



Profiling time and performance of ProbData over an emulated 10Gbps connection of 380ms RTT between ANL and UChicago.



Thanks ! ☺

Questions ?