Quality of Service Aspects and Metrics in Grid Computing

Daniel A. Menascé
Dept. of Computer Science
George Mason University
Fairfax, VA
menasce@cs.gmu.edu

Emiliano Casalicchio
Dipt. Informatica Sistemi e Produzione
Univ. Roma “Tor Vergata”
Rome, Italy
casalicchio@ing.uniroma2.it

Outline

- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

Outline

- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

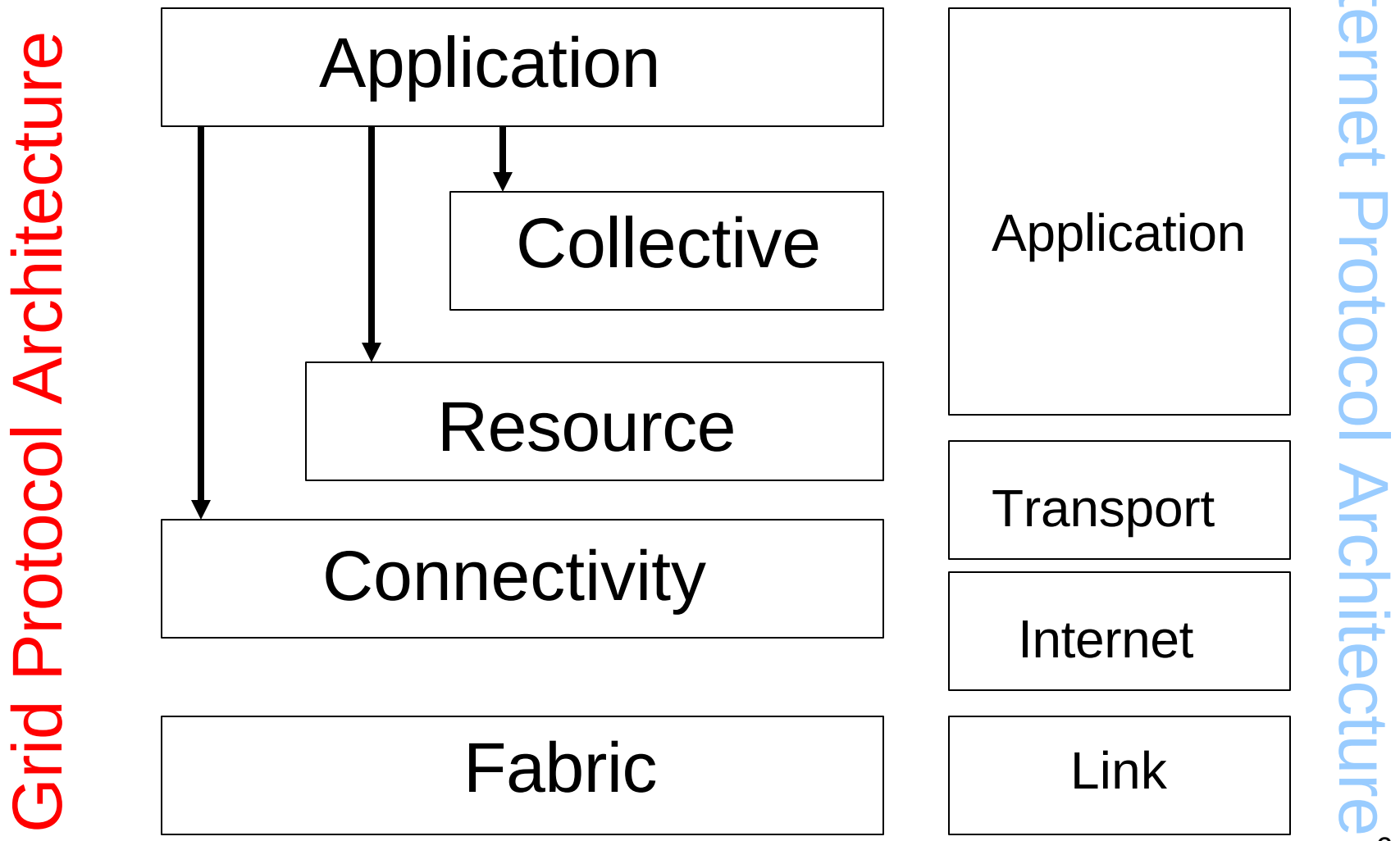
What is Grid Computing?

- Hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.
- Main paradigm for compute intensive scientific applications but is now becoming an important model for commercial applications.
- Grid computing is about sharing resources (computing, storage, data storage, and service providers) in a transparent manner with good Quality of Service.

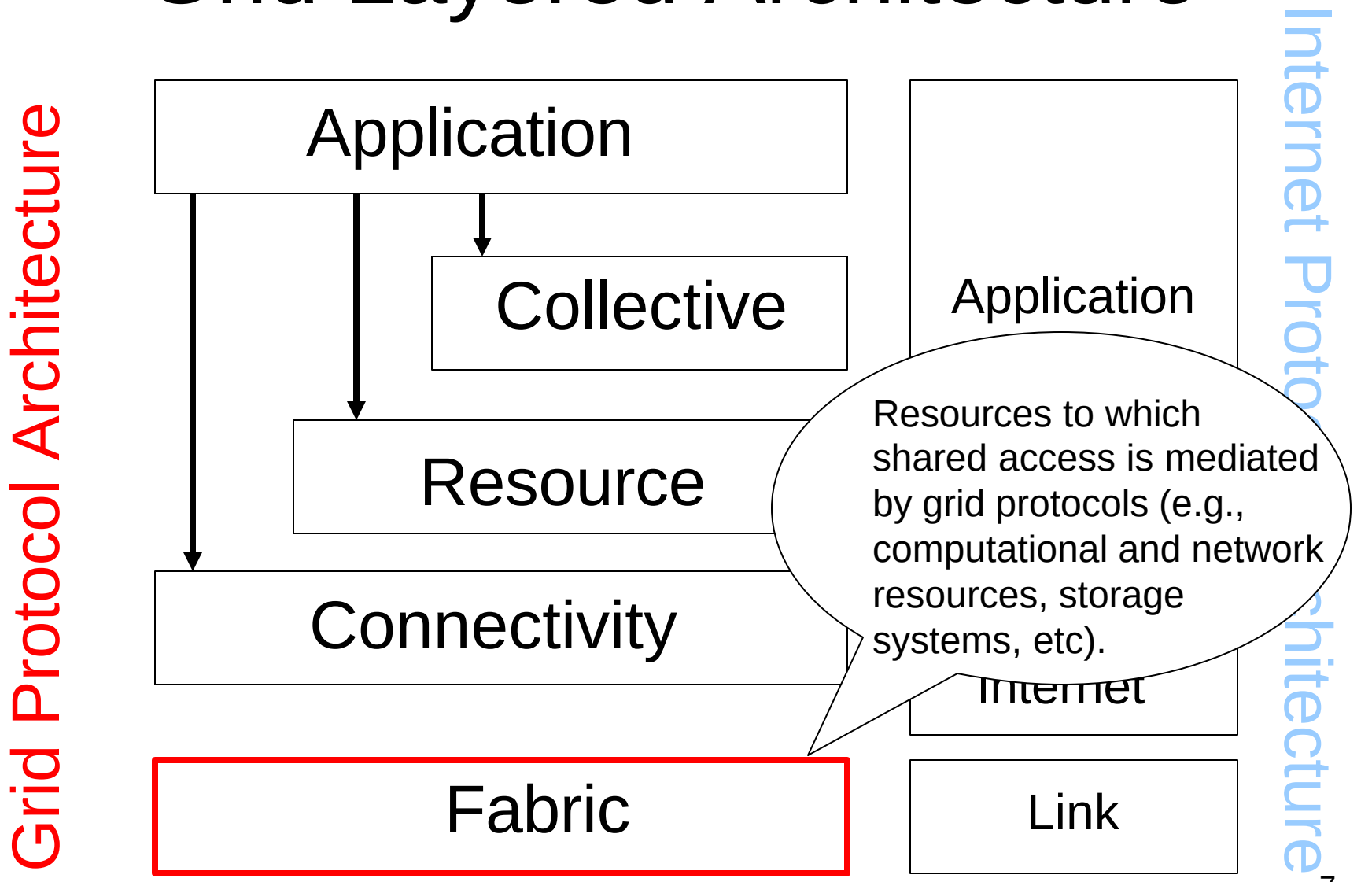
Virtual Organizations (VOs)

- Provide non trivial QoS: faster response to changing business needs, better utilization and service level performance, and lower IT operating costs.
- Integration and coordination of resources that belong to different domains characterized by different resource management and security policies.

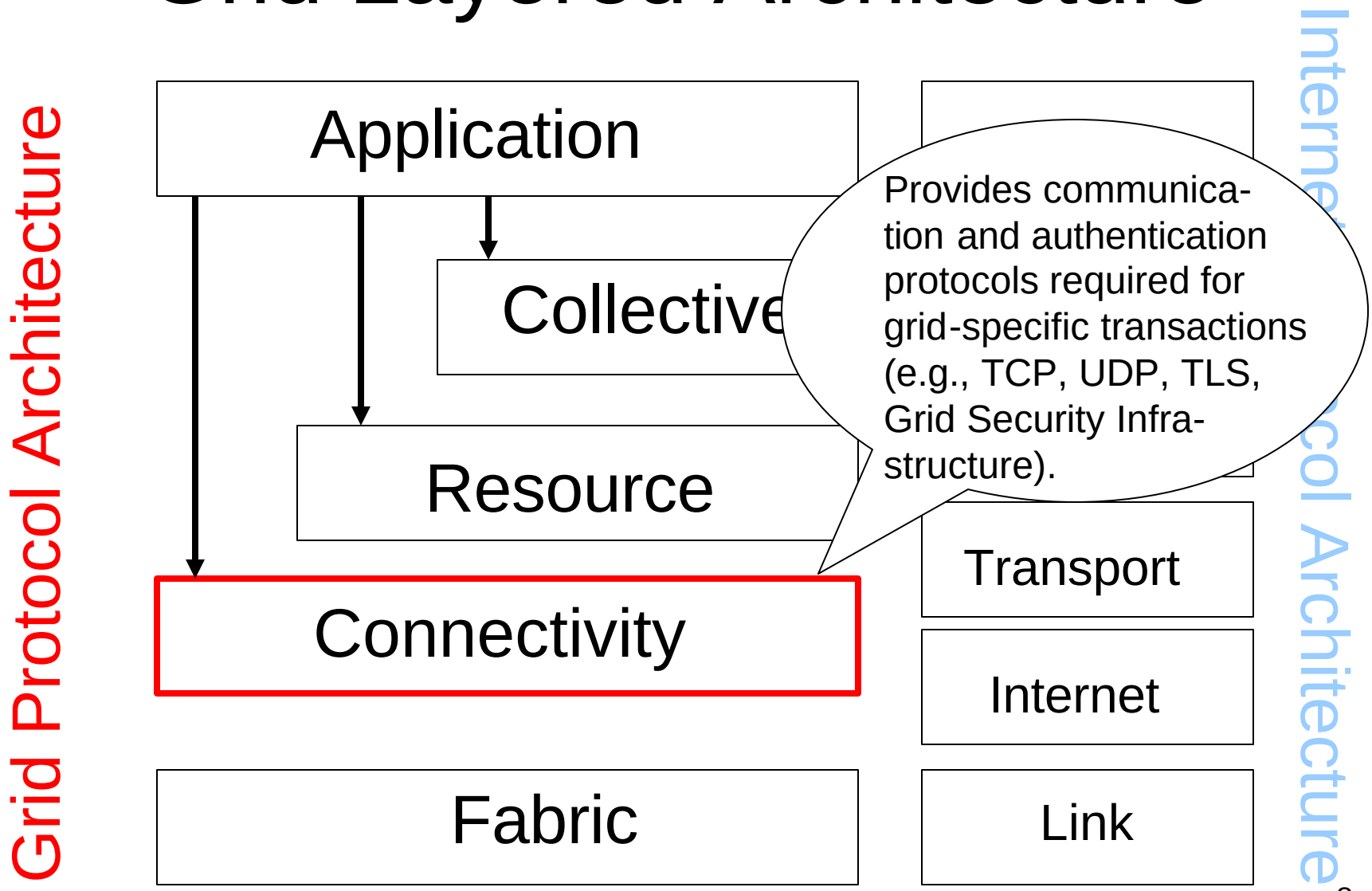
Grid Layered Architecture



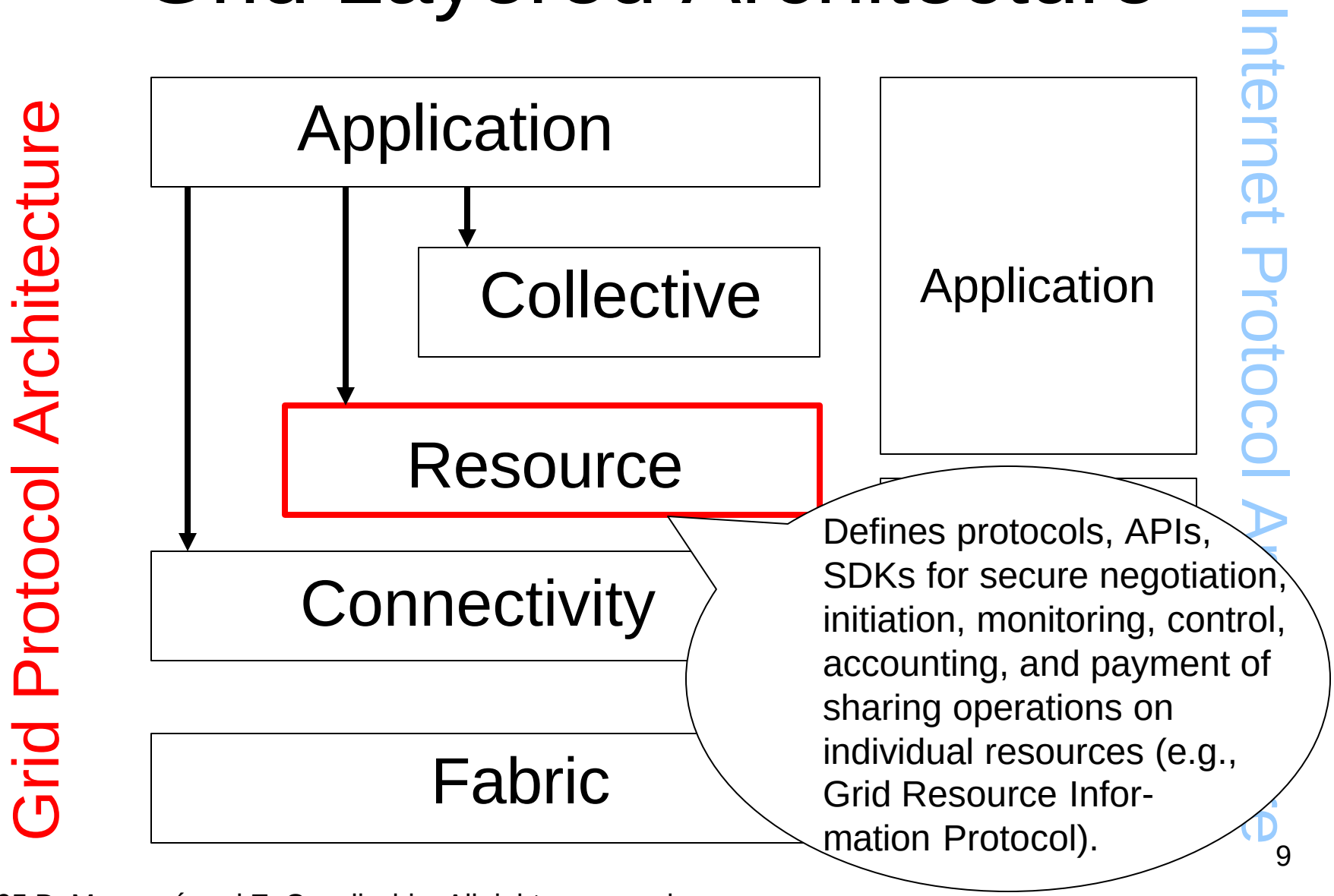
Grid Layered Architecture



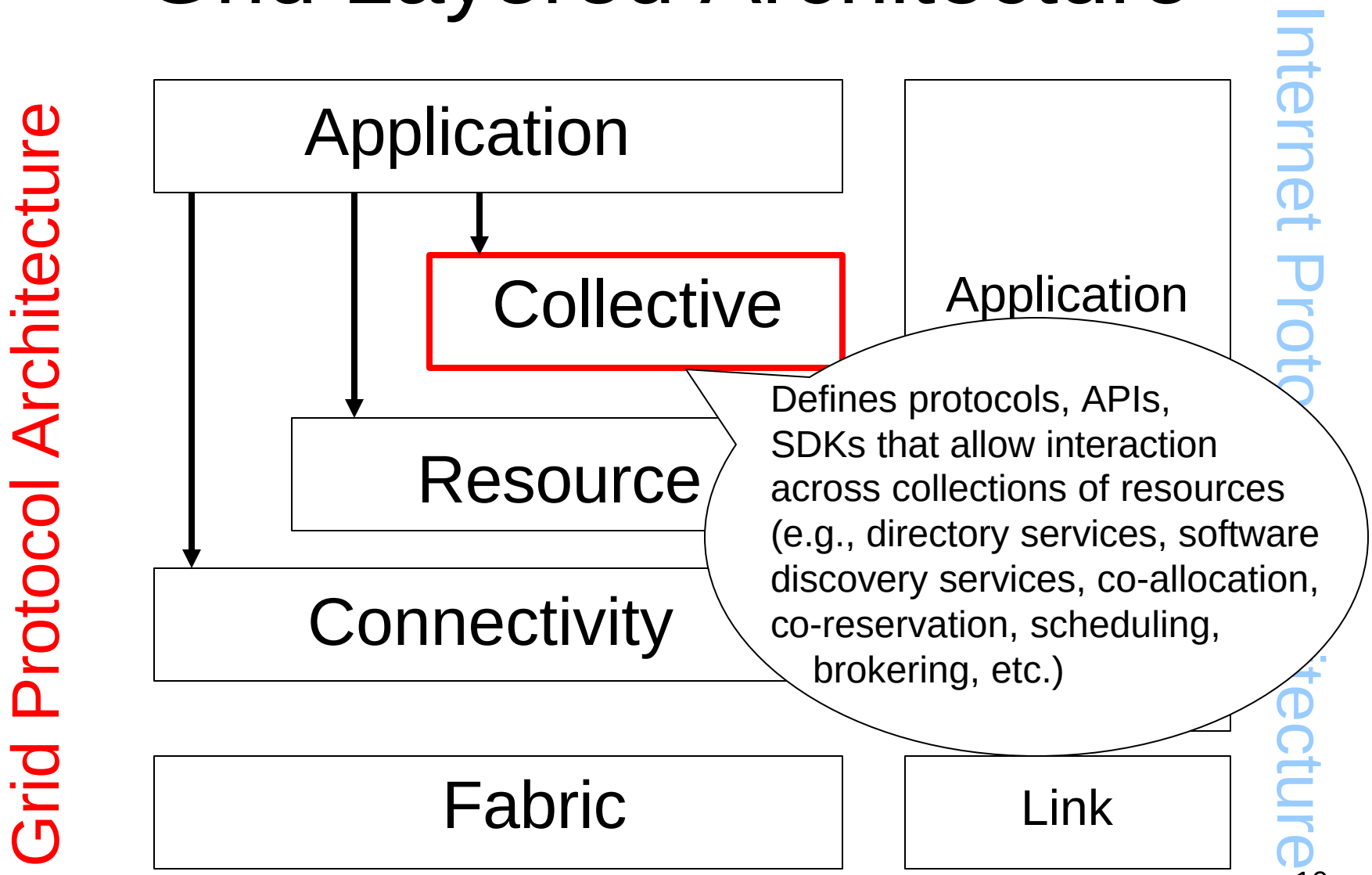
Grid Layered Architecture



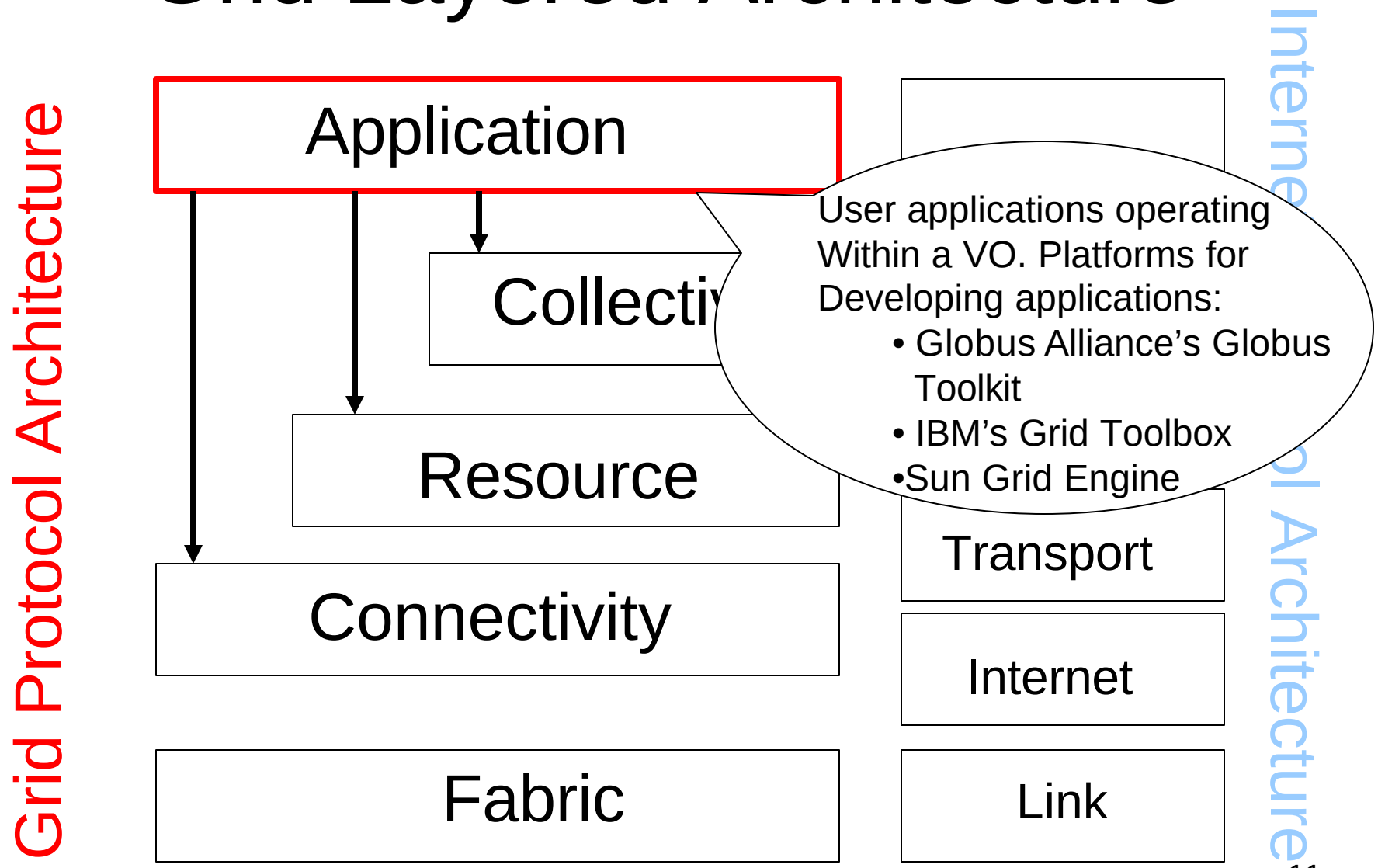
Grid Layered Architecture



Grid Layered Architecture



Grid Layered Architecture



Grid Portal Development Kit (GPDK)

The GPDK is not currently being supported anymore!

The GPDK is not currently being supported anymore!

- Introduction
- Architecture
- Grid Service Beans
- Download
- Resources

Introduction

One of the most exciting areas of Grid application development is the construction of portals to allow computational scientists, researchers and high performance computer/application users access to resources via an easy to use web page interface. The goal is to develop common components that can be used by portal developers to build a website that can securely authenticate users to remote resources and help them make better decisions for scheduling jobs by allowing them to view pertinent resource information obtained and stored on a remote database. In addition, profiles are created and stored for portal users allowing them to track and monitor jobs submitted and view results.

The Grid Portal Development Kit provides both a portal development environment for the creation of new portals as well as a collection of Grid service beans used to accomplish basic operations such as job submission, file transfer and querying of information services.

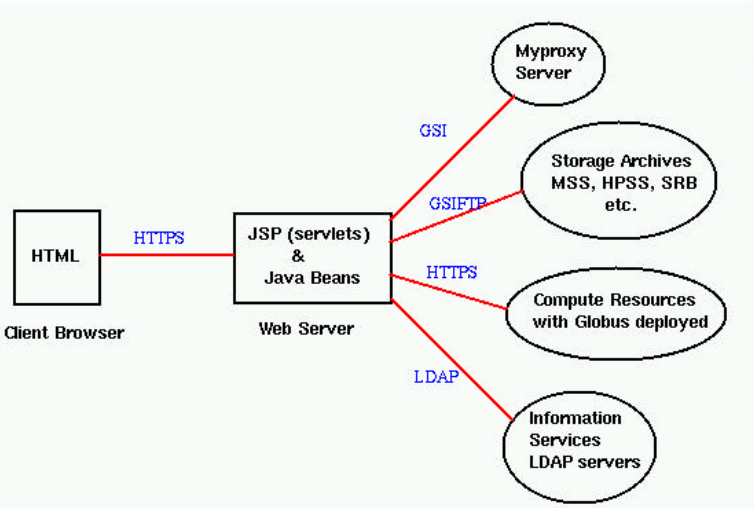
Architecture

The envisoned portal architecture is a standard three tier model, where a client browser securely communicates to a web server over an https connection. The web server is capable of accessing various Grid services using the Globus infrastructure. The Globus toolkit provides mechanisms for securely submitting jobs to a Globus gatekeeper, querying for hardware/software information using LDAP, and a secure PKI infrastructure using GSI.

By taking advantage of the Myproxy package , also distributed as part of the Grid Portal Collaboration, users can use the portal to gain access to remote resources from anywhere without requiring their certificate and private key be located on the same machine/device running a web browser. The Myproxy server is responsible for maintaing user's delegated credentials, proxies, that can be securely retrieved by a portal for later use.

The Grid Portal Development Kit is designed to provide access to Grid services by using Java Server Pages (JSP) and Java Beans using Tomcat, an open source web application server developed by Sun microsystems as the latest reference implementation of Java Servlets v2.2 and Java Server Pages v1.1. Tomcat can be easily configured to work with an existing web server e.g. Apache, Microsoft IIS, or Netscape as well as its own simple web server that can be used for development purposes. The GPDK is packaged as a web application as defined by the Java Servlet 2.2 specification, consisting of web pages (HTML), Java server pages (JSP), and Java beans. A web application can be easily packaged and deployed as a Web application ARchive (WAR) file similar to a Java ARchive file containing Java classes.

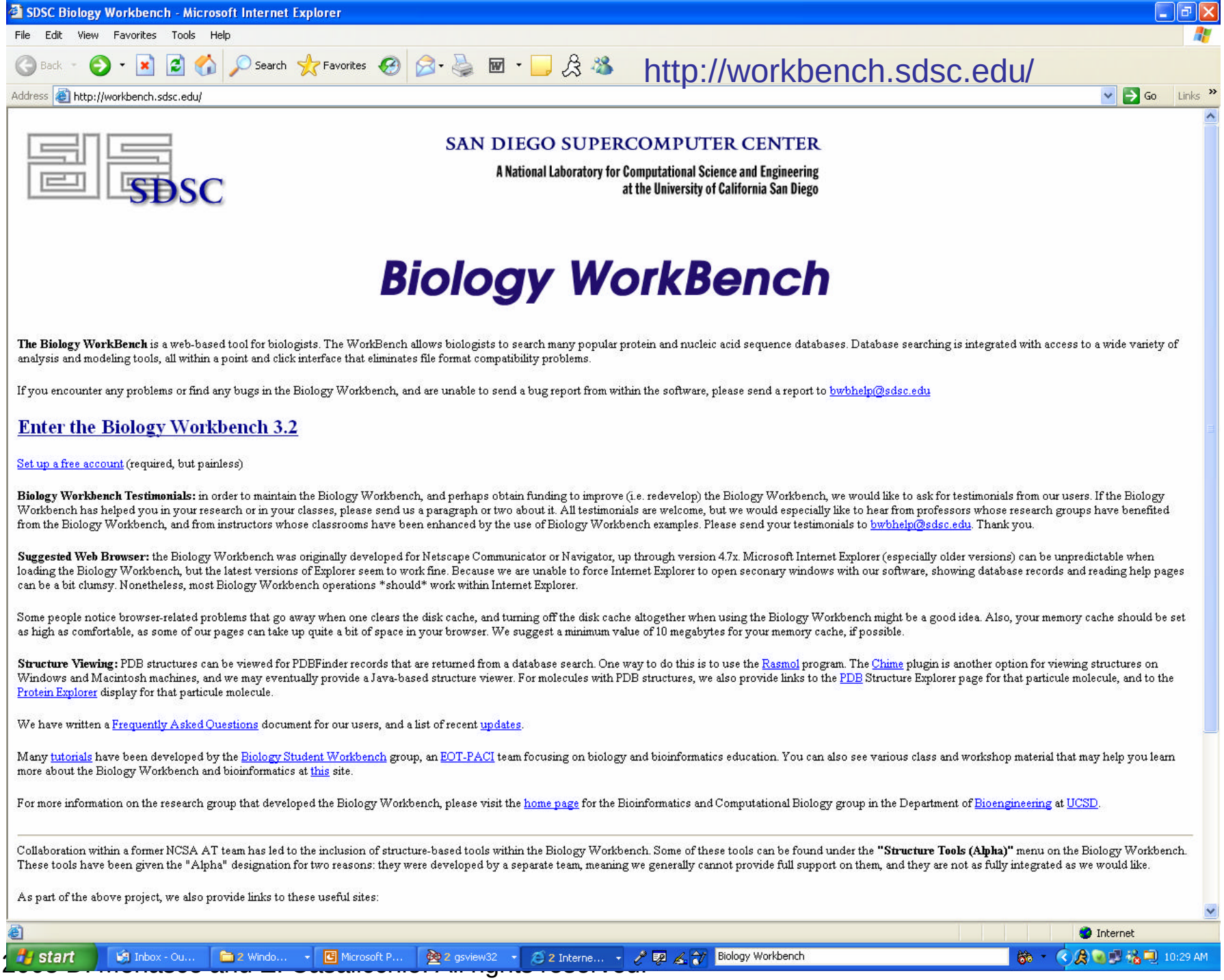
The GPDK Java beans derive most of their functionality from the Globus Java Commodity Grid (CoG) toolkit. CoG provides a Globus API in pure Java including the GSI using the IAIK Java SSL libraries to delegate credentials. The CoG kit provides API's for submitting jobs to Globus gatekeepers, transferring files using GSIFTP implemented in Java and querying LDAP servers using the Java Naming and Directory Interface (JNDI). GPDK Java beans merely present an easier interface for web developers to use the CoG kit when developing portal server pages.



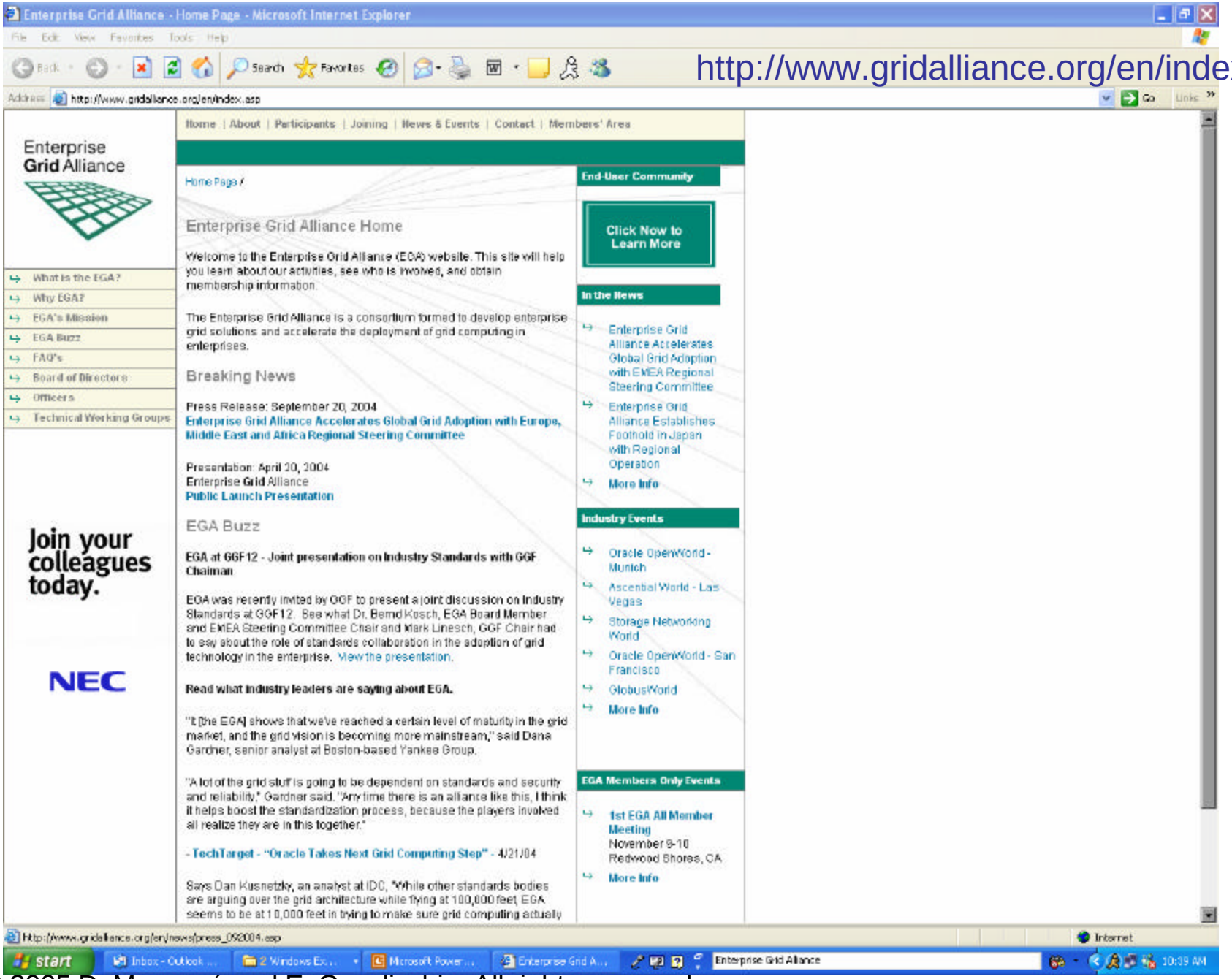
Grid Service Beans

GPDK Java Beans are grouped into the following five categories, discussed below: Security, User Profiles, Job Submission, File Transfer and Information Services.

Security




FightAIDS@Home - Download - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://fightaidsathome.scripps.edu/download-FAAH.html
Go Links
http://fightaidsathome.scripps.edu/download-FAAH.html
fightAIDS@home
powered by AUTODOCK
The Olson Laboratory
The Scripps Research Institute
Member Login
Fight AIDS @ Home
The AIDS Crisis
How Your PC can Help
Project Status
Get the Download
Research Team
The Discovery Process
Links and Communities
Link Your Site to FA@H
FAQ
Download FightAIDS@Home
Joining FightAIDS@Home is easy!
When you click on the download link below, the software for FightAIDS@Home will be installed on your computer. During the installation process, you will be asked to create your own password for your FightAIDS@Home membership. Be sure you write down both your MemberID and Password.
Once the software is installed,a new browser window will open where you will be asked to type in your password to start your participation in FightAIDS@Home. It is important that you do not close this browser window until you have typed in your password and clicked "Submit" to activate your participation.
It's that simple. After you have typed in and submitted your password, your computer will automatically "call in for a homework assignment" when you connect to the Internet. Once that assignment is complete, your computer will check to see if you are online, and if you are, it will turn in the "homework" assignment and ask for another.
System Requirements for FightAIDS@Home:
Windows XP, 2000, NT 4.0, Me, or 98
Internet Explorer 5.x installed
96MB Memory (RAM) minimum
Pentium 133 or above
Note: In Phase 2, we plan to be able to support in the future these additional platforms:
Mac OS X
Linux
These are not available yet, nor will they be around any time soon. But, stay tuned for news of new releases on these platforms.
Read the FightAIDS@Home Software Usage Policies
Done
Internet
start
Inbox - Ou...
2 Windo...
Microsoft P...
2 gsview32
3 Interne...
FightAIDSatHOME
10:33 AM



Outline

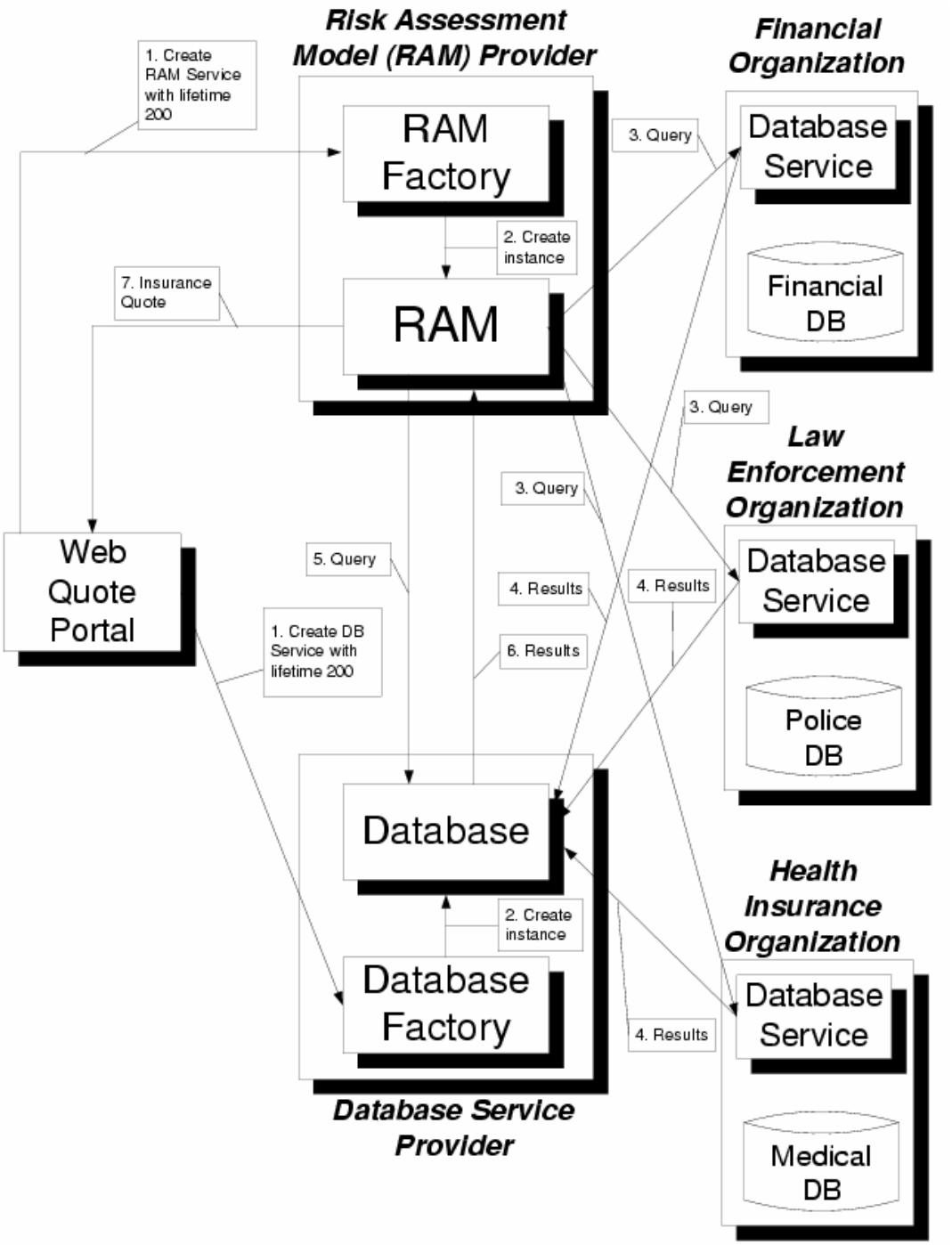
- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

Motivating Example

- An insurance company wants to increase its profit by minimizing its risk and attracting/retaining more customers.
- Traditionally, insurance companies use risk assessment models (RAMs) that place customers into very broad categories:
  - All non-smoking male straight-A college students under the age of 25.
- Our insurance company wants to use customized RAMs:
  - John Doe who is a non-smoking male straight-A student at George Mason University, is 24 years old, is a member of the Association for Computing Machinery, a member of a programming team that won a regional prize in an ACM-sponsored programming contest, has a very good credit record, undergoes a physical exam every year and is in perfect health, and has a clean record with federal, state, and local law enforcement agencies.

Motivating Example (cont'd)

- The IC plans to harness the resources in its worldwide network of computers to provide the infrastructure to run the customized models.
- Three types of insurance quotes:
  - Immediate: use simple RAMs and return results in a few seconds
  - Non-immediate: return results in a few minutes while the user is still online and uses more complex RAMs
  - Delayed: may take hours to process and uses very sophisticated RAMs. Results provided by e-mail.



Outline

- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

QoS Metrics in a Grid Environment

- Latency: time to execute a task.
  - Application layer:
    - Average time to complete immediate quote requests
    - 95-percentile of the time to respond to non-immediate queries
    - Elapsed time to return a delayed quote
  - Collective layer:
    - Average time to perform co-reservation and co-allocation of resources to run a RAM
  - Resource layer:
    - Time to access a specific resource through the Grid Resource Access Management protocol.
  - Connectivity layer:
    - Time to perform an authentication on behalf of a customer to a financial organization using the GSI.
  - Fabric layer:
    - Time taken by the database server at a health insurance company to reply to a query.

QoS Metrics in a Grid Environment

- Throughput: units of work executed per unit time.
  - Application layer:
    - Number of delayed requests processed per second.
  - Collective layer:
    - Number of queries per second that can be handled by the directory service used to locate resources across different VOs.
  - Resource layer:
    - Effective transfer rate (in KB/sec) under the Grid-FTP protocol used to transfer files among computers running RAMs.
    - Queries/sec that can be processed by the database server of a law enforcement agency needed by a RAM application
  - Connectivity layer:
    - Throughput (in KB/sec) of a secure connection between an instance of a RAM model and a database service that provides information about customers.
  - Fabric layer:
    - Number of CPU cycles per second obtained from a machine involved in processing a task that is part of a parallel RAM evaluation.

QoS Metrics in a Grid Environment

- Availability: fraction of time a resource/application is available for use
  - Application layer:
    - Fraction of time that the immediate quote requests application is available.
  - Collective layer:
    - Fraction of a time that the directory service used to locate resources across different VOs is available.
  - Resource layer:
    - Fraction of time that a specific computing node is available for allocation to a RAM instance.
  - Connectivity layer:
    - Fraction of time that a proxy authentication request succeeds in authenticating a user with a law enforcement agency.
  - Fabric layer:
    - Fraction of time that a computing node is available during the execution of a RAM instance.

Metrics Across Layers

- Elapsed time to process a delayed quote request depends on:
  - Time to reserve computing cycles for the duration of the RAM evaluation
  - Time to co-allocate resources
  - Time to authenticate with databases outside the insurance company
  - Time to obtain inputs from external databases
  - Time to transfer data files to different nodes of a parallel job running a RAM
  - Time to execute the RAM

Outline

- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

Grid Resource Allocation and SLAs

- Problem: map application-level SLAs to lower-level SLAs.
  - How to select service providers and the services within these service providers in a way that global SLAs are met at minimum cost?

Grid Resource Allocation and SLAs

- RAM evaluation requires NC million CPU cycles and has to be completed in at most T_{max} time units and its computational cost cannot exceed C_{max} dollars.
- There are three available computing resources with speed s_i in 10^6 cycles/sec and cost c_i in dollars/sec.
- Possible co-allocations: (1), (2), (3), (1,2), (1,3), (1,3), and (1,2,3)

Grid Resource Allocation and SLAs

- N: number of computing resources
- NC_i: number of cycles allocated to computing resource i.
- T: execution time of a given allocation
- C: cost of an allocation.

Constraints:

$$\sum_{i=1}^{N} NC_i = NC$$

$$T = \max_{i=1}^{N} \left\{ \frac{NC_i}{s_i} \right\} \leq T_{\max}$$

$$C = \sum_{i=1}^{N} \frac{NC_i}{s_i} \times c_i \leq C_{\max}$$

Grid Resource Allocation and SLAs

- If there are no cost constraints, the solution that minimizes the total execution time is the one that allocates more cycles to the faster resources in proportion to its speed:

$$NC_i = NC \times \frac{s_i}{\sum_{j=1}^{N} s_j}$$

Grid Resource Allocation Optimization Problems

- Problem 1 (execution time minimization): “Find the feasible solution that satisfies the cost constraint at minimum execution time.”

- Problem 2 (cost minimization): “Find the feasible solution that minimizes the cost C and that satisfies the execution time constraint.”

Execution Time Minimization

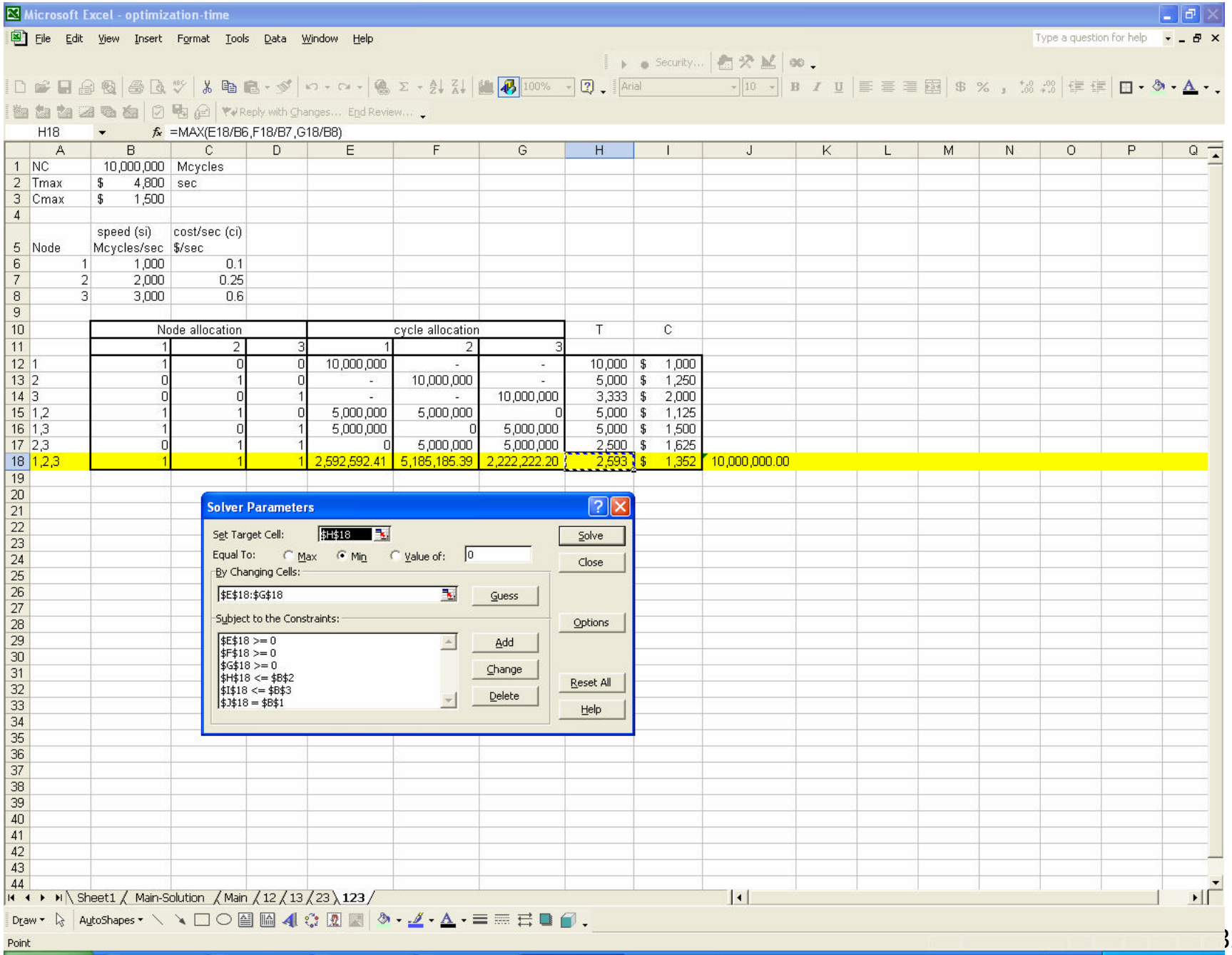
Inputs:

Node	speed (si) Mcycles/sec	cost/sec (ci) $/sec
1	1,000	0.1
2	2,000	0.25
3	3,000	0.6

NC		10,000,000	Mcycles
Tmax	$	4,800	sec
Cmax	$	1,500	

Solution: Obtained using MS Excel's Solver Tool.

Allocation	Cycle Allocation (NCi)			T	C
1	10,000,000	-	-	10,000	1,000
2	-	10,000,000	-	5,000	1,250
3	-	-	10,000,000	3,333	2,000
1,2	3,333,333	6,666,667	0	3,333	1,167
1,3	4,800,000	0	5,200,000	4,800	1,520
2,3	0	6,666,667	3,333,333	3,333	1,500
1,2,3	2,592,592.41	5,185,185.39	2,222,222.20	2,593	1,352



Cost Minimization

Inputs:

Node	speed (si) Mcycles/sec	cost/sec (ci) $/sec
1	1,000	0.1
2	2,000	0.25
3	3,000	0.6

NC		10,000,000	Mcycles
Tmax	$	4,800	sec
Cmax	$	1,500	

Solution: Obtained using MS Excel's Solver Tool.

	Cycle Allocation (NCi)			T	C
1	10,000,000	-	-	10,000	1,000
2	-	10,000,000	-	5,000	1,250
3	-	-	10,000,000	3,333	2,000
1,2	4,800,000	5,200,000	0	4,800	1,130
1,3	5,000,000	0	5,000,000	5,000	1,500
2,3	0	9,600,000	400,000	4,800	1,280
1,2,3	4,800,000	5,200,000	-	4,800	1,130

Comparing the Time and Cost Optimization Solutions

	Time Optimization		Cost Optimization	
Allocation	%Tmax	%Cmax	%Tmax	%Cmax
(1,2)	69	78	100	75
(2,3)	69	100	100	85
(1,2,3)	54	90	90	75

Outline

- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

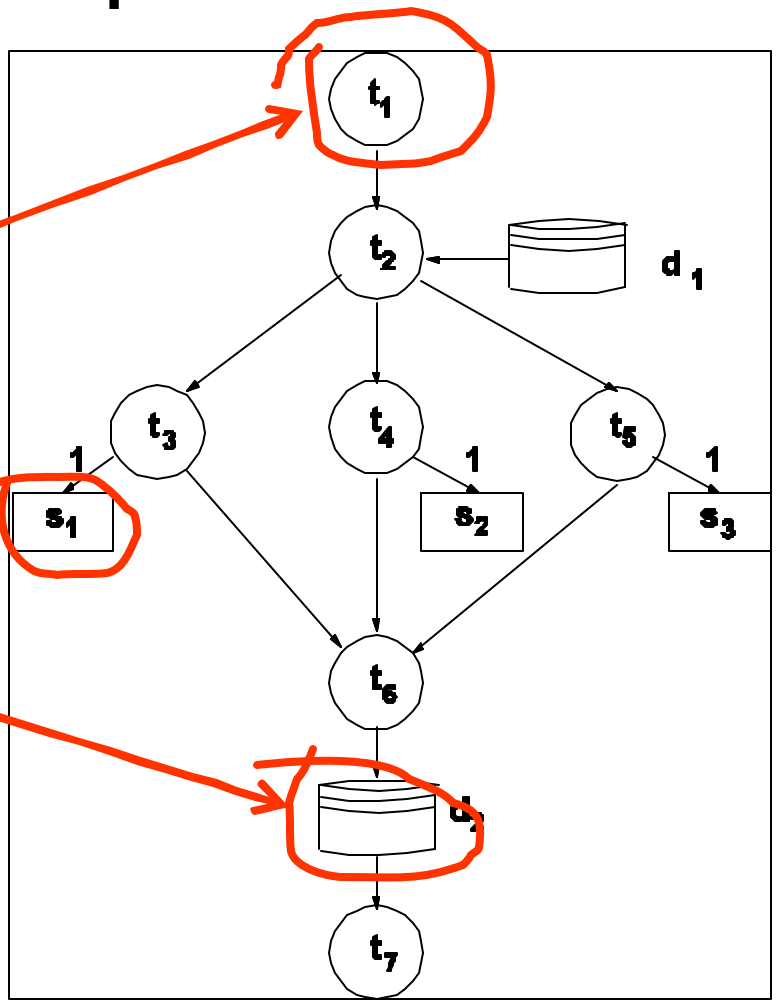
Assumption on Resources

- Resources are **shared** by more than one grid application
- Four classes of resources are considered:
  - **Compute resources**: provide cycles to computations
    - (e.g., RAM evaluation).
  - **Service Provider resources**: provide services upon request
    - (e.g., access to financial, law enforcement, and health insurance databases).
  - **Network resources**: provide the bandwidth necessary for nodes of a grid to communicate and transfer data.
  - **Data Storage resources**: provide storage for data generated during the computation of a grid application.
- **Several instances** of a resource of a given class
  - e.g., several SPs that provide access to law enforcement databases
  - e.g., SP nodes may offer the same functionality with different SLA and at different costs.

Task Graph

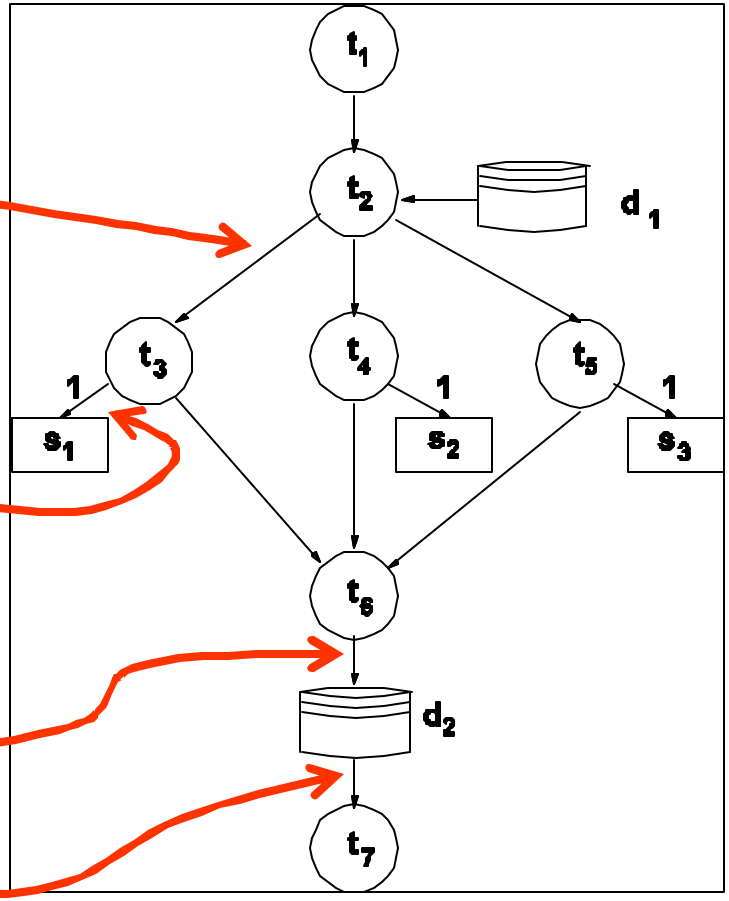
An application is formally specified by a task graph

- Three type of nodes
  - *Computation task nodes* specify nodes where computation is performed
  - *Service providing nodes* indicate the invocation of a service by a computation task.
  - *Logical data storage nodes* denote data stores used by tasks. These logical data storage elements have to be mapped into physical data storage devices, which may be accessed over a network.

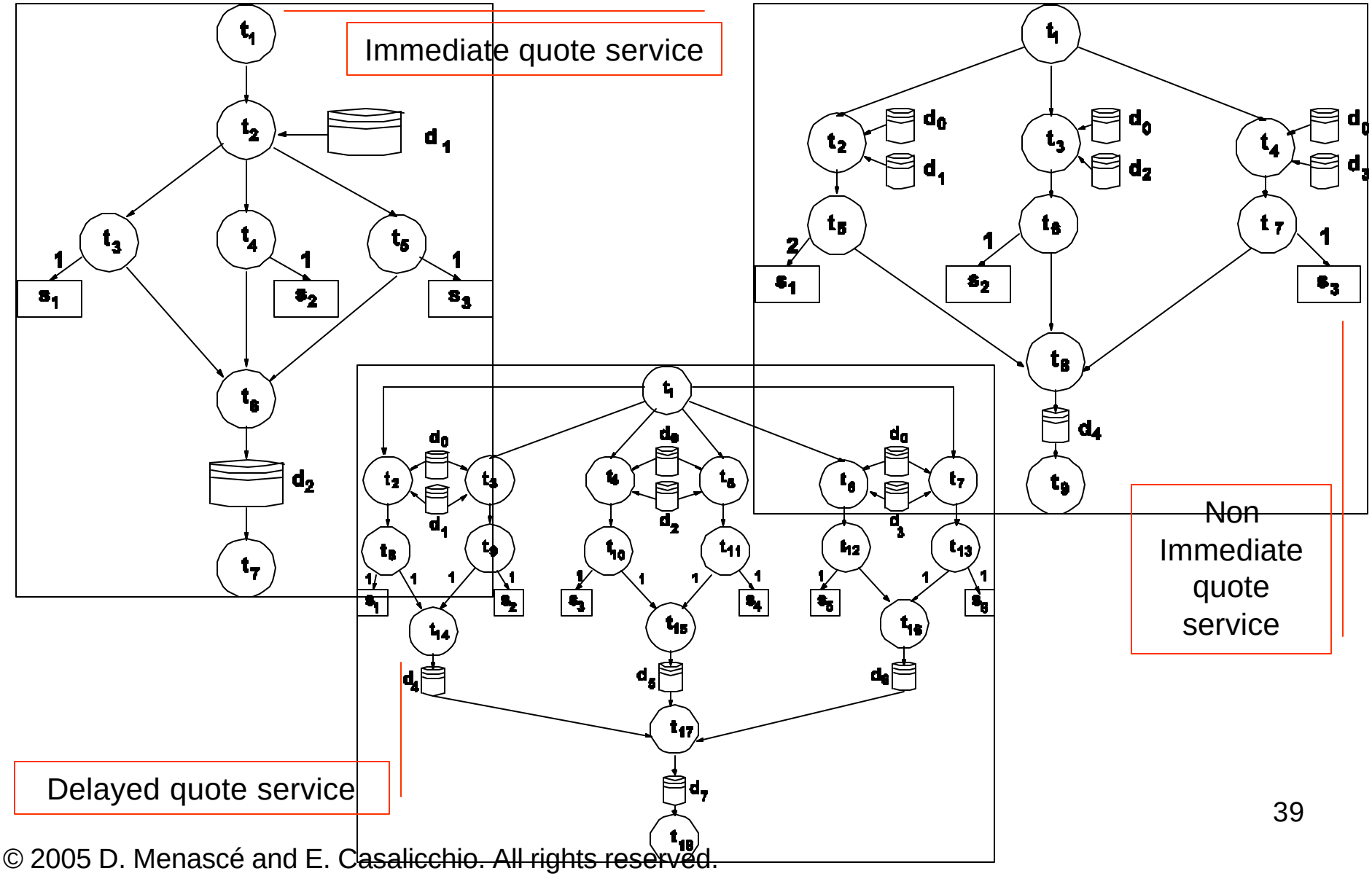


Task Graph: type of arcs

- *Precedence arc.* From computation task ti to task tj
  - indicates that tj can only start after ti finishes and after ti has transmitted all the data it needs to send to tj .
- *Service request arc.* From a computation task node to a service providing node
  - indicates that task ti invokes service sj during its execution.
- *Data storage arc.* From task to data storage nodes and vice-versa
  - indicates that task ti generates data storage dk.
  - indicates that ti uses data from storage element dk and therefore cannot start until the data storage is generated.



Task Graph: Examples



Formal Model

- Application related parameters
- Resource related parameters
- Cost related parameters

Application Related Parameters

- $\mathcal{T}$: set of tasks of the application,
- T: execution time, in sec, of an application,
- NC_i: number of compute cycles, in millions of cycles, required by task i,
- C_i: execution time, in sec, of task i,
- $V_{i,s}$: average number of times that task i invokes logical service provider s,
- $x_{i,j}$: total amount of data, in Mbits, transferred from task i to task j,
- $T_{i,j}^k$: network time, in seconds, needed by task i to send data to task j over network resource k,
- $w_{i,m}$: amount of data, in Mbytes, transferred (i.e., written) from task i to logical data storage m,
- $r_{m,i}$: amount of data, in Mbytes, transferred (i.e., read) by task i from logical data storage m.
- $W_{i,m}$: time, in seconds, needed by task i to write into logical data storage m,
- $R_{m,i}$: time, in seconds, needed by task i to read from logical data storage m,
- N_{LDS}: number of logical data storage nodes used by the application, and
- N_{LSP}: number of logical service providers used by the application.

Resource Related Parameters

- $\mathcal{C}$: set of computing resources. $|\mathcal{C}| = C$,
- $\mathcal{S}$ set of available physical service providers. $|\mathcal{S}| = S$,
- $\mathcal{N}$: set of available network resources. $|\mathcal{N}| = N$,
- $\mathcal{D}$: set of physical data storage resources. $|\mathcal{D}| = D$,
- s_k: speed, in millions of cycles per second, of computing resource k $(k = 1, \cdots, C)$,
- R_k: response time, in seconds, of a service request at service provider k $(k = 1, \cdots, S)$,
- B_k: bandwidth, in Mbps, of network resource k $(k = 1, \cdots, N)$,
- L_k: latency, in seconds, of network resource k $(k = 1, \cdots, N)$, and
- X_k: transfer rate, in Mbytes/sec, of physical data storage resource k $(k = 1, \cdots, D)$.

Cost Related Parameters

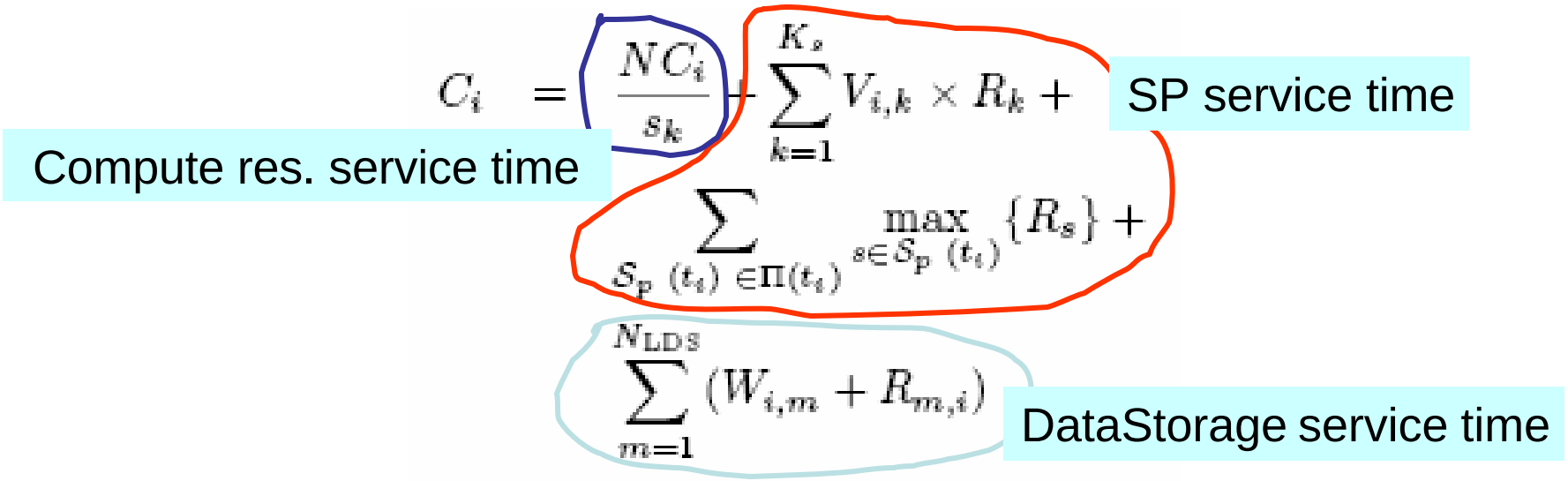
- c_k^c: cost, in \$/sec, per second of usage of compute resource k $(k = 1, \cdots, C)$,
- c_k^b: cost, in \$/Mbps, per unit of bandwidth measured in Mbps provided by network resource k $(k = 1, \cdots, N)$,
- c_k^d: cost, in \$/Mbyte, for each Mbyte of data transferred to/from data storage resource k $(k = 1, \cdots, D)$, and
- c_k^s: cost, in dollars, for service provider k $(k = 1, \cdots, S)$ to fulfill a service request. Physical service providers that provide the same functionality with different response time SLAs are considered different resources.

Execution Time of a Grid Application

- ***T*** is a function of three components**:**
  - *Task execution time.* The execution time of task ti when running on computation resource k.
  - *Data transfer time.* The write and read times for a task on a data storage m
  - *Network time.* The network time associated with the transmission of data between tasks over a network resource

Task Execution Time

- The execution time of task t_i when running on computation resource k is given by



  - Ks is the number of services sequentially invoked by task ti,
  - Sp(ti) is a set of services invoked in parallel by task ti,
  - Π(ti) is the set of sets of parallel service invocations of task ti.

Data Transfer Time

- The write and read times for task *i* on data storage *m*, when it is mapped to physical data storage resource *k* are given by

$w_{i,m}$: amount of data, in Mbytes, transferred (i.e., written) from task i to logical data storage m,

$$W_{i,m} = \frac{w_{i,m}}{X_k}$$

X_k: transfer rate, in Mbytes/sec, of physical data storage resource k ($k = 1, \cdots, D$).

$r_{m,i}$: amount of data, in Mbytes, transferred (i.e., read) by task i from logical data storage m.

$$R_{m,i} = \frac{r_{m,i}}{X_k}$$

Network Time

- The network time $T^k_{i,j}$ associated with the transmission of $x_{i,j}$ Mbits of data between tasks *i* and *j* over network resource *k* is

B_k: bandwidth, in Mbps, of network resource k ($k = 1, \cdots, N$),

$$T^k_{i,j} = L_k + x_{i,j}/B_k.$$

L_k: latency, in seconds, of network resource k ($k = 1, \cdots, N$), and

Logical to Physical Mapping

- Four mappings are required to compute the execution time ***T***
  - tasks to computing resources (TC),
  - communication patterns to network resources (CN),
  - service providers to physical service providers (SPA),
  - data storage elements to physical storage elements (ST).
- formally, a mapping is ***M*=(TC, CN, SPA, ST)**

The Optimization Problem

- The optimization problem to be solved is:
  - given a grid application *A*, find the mapping *M* that minimizes the cost *C(A,M)* while satisfying the execution time *SLA* of *T<=Tmax*.

- The cost *C(A,M)* of running application A using mapping *M* is given by

$$C(A,\mathcal{M}) = \sum_{i=1}^{|\mathcal{T}|}\sum_{k=1}^{C} TC[i,k] \,.\, \frac{NC_i}{s_k} \,.\, c_k^c +$$ Computing cost

$$\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{|\mathcal{T}|} c^b_{CN[i,j]} \,.\, B_{CN[i,j]} +$$ Networking cost

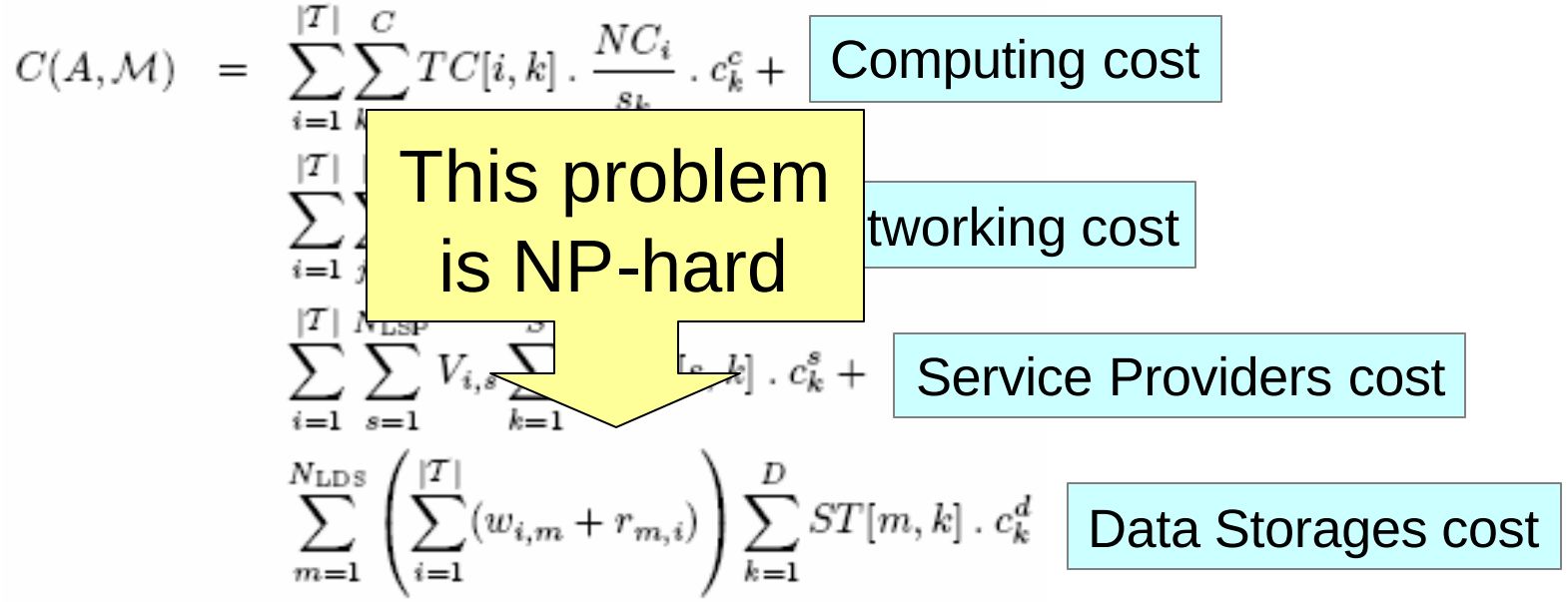
$$\sum_{i=1}^{|\mathcal{T}|}\sum_{s=1}^{N_{\mathrm{LSP}}} V_{i,s} \sum_{k=1}^{S} SPA_i[s,k] \,.\, c_k^s +$$ Service Provider cost

$$\sum_{m=1}^{N_{\mathrm{LDS}}} \left(\sum_{i=1}^{|\mathcal{T}|} (w_{i,m} + r_{m,i}) \right) \sum_{k=1}^{D} ST[m,k] \,.\, c_k^d$$ Data Storage cost

The Optimization Problem

- The optimization problem to be solved is:
  - given a grid application *A*, find the mapping *M* that minimizes the cost *C(A,M)* while satisfying the execution time *SLA* of *T<=Tmax*.

- The cost *C(A,M)* of running application A using mapping *M* is given by

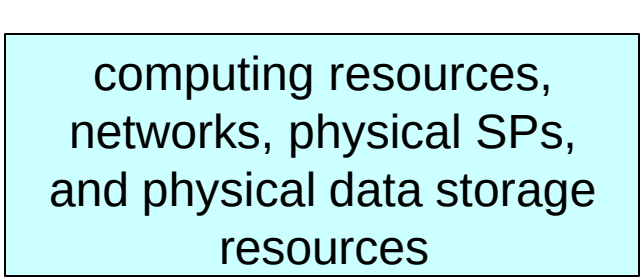


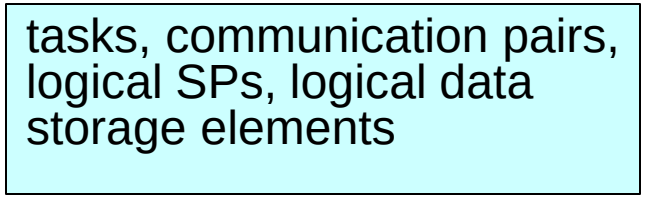
Outline

- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

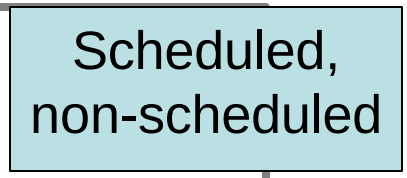
Heuristics

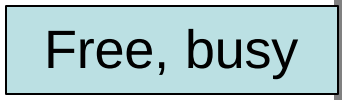
- A multi-resource assignment problem
- The scheduling algorithm (DES based) consists of:
  - a loop that is executed until all logical entities are allocated to physical resources





Loop
Domain selection
T+: task domain,
C+, S+, N+, D+: resources (compute, SP, network and storage) domain
Heuristics
Domain Update
End Loop





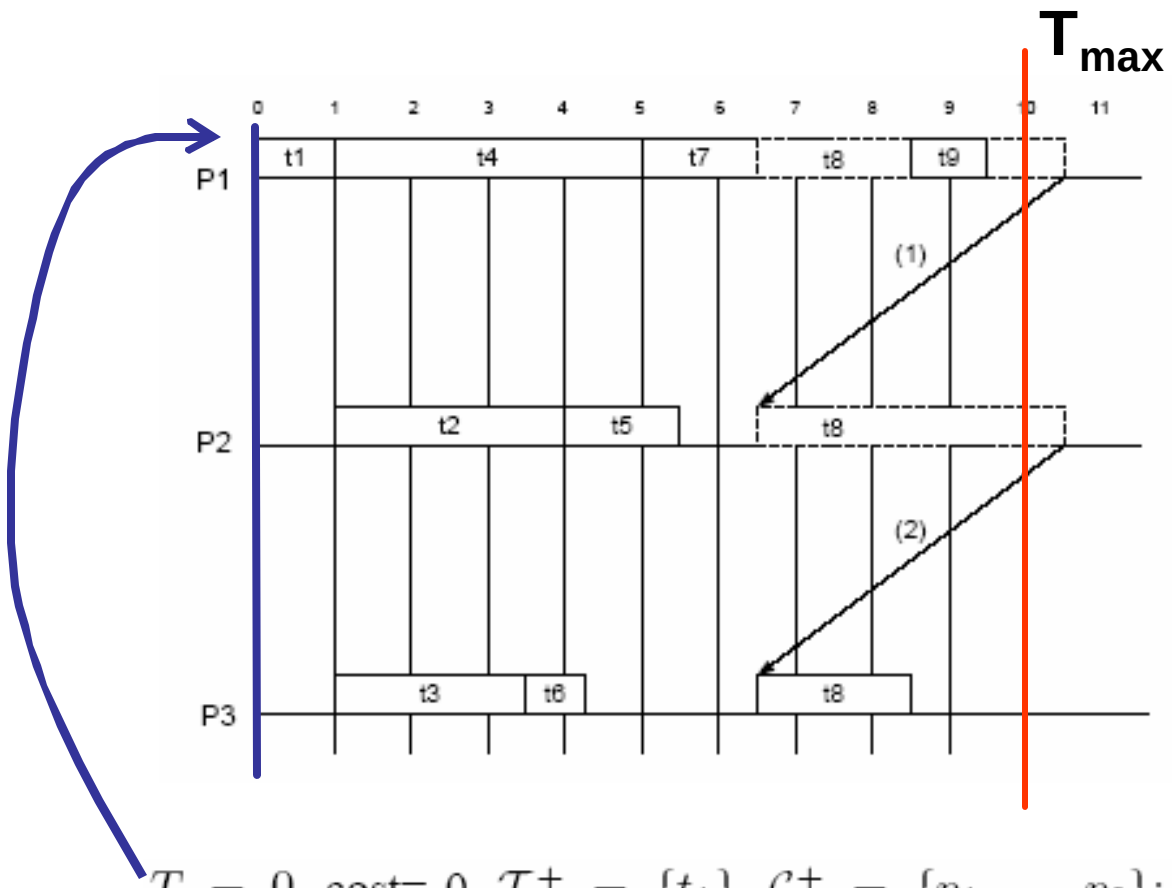
Heuristic

The following two rules explain the heuristic used at an **allocation instant**.

1. If |T+|=1, then assign the task in T+ to the least expensive computing resource in C+.
2. If |T+|> 1 then the tasks in T+ are allocated to the |T+| least expensive computing resources in C+. The allocation of tasks to this subset of C+ is done in such a way that tasks with the largest computational demand (i.e., largest number of cycles) are allocated to the fastest computing resources.

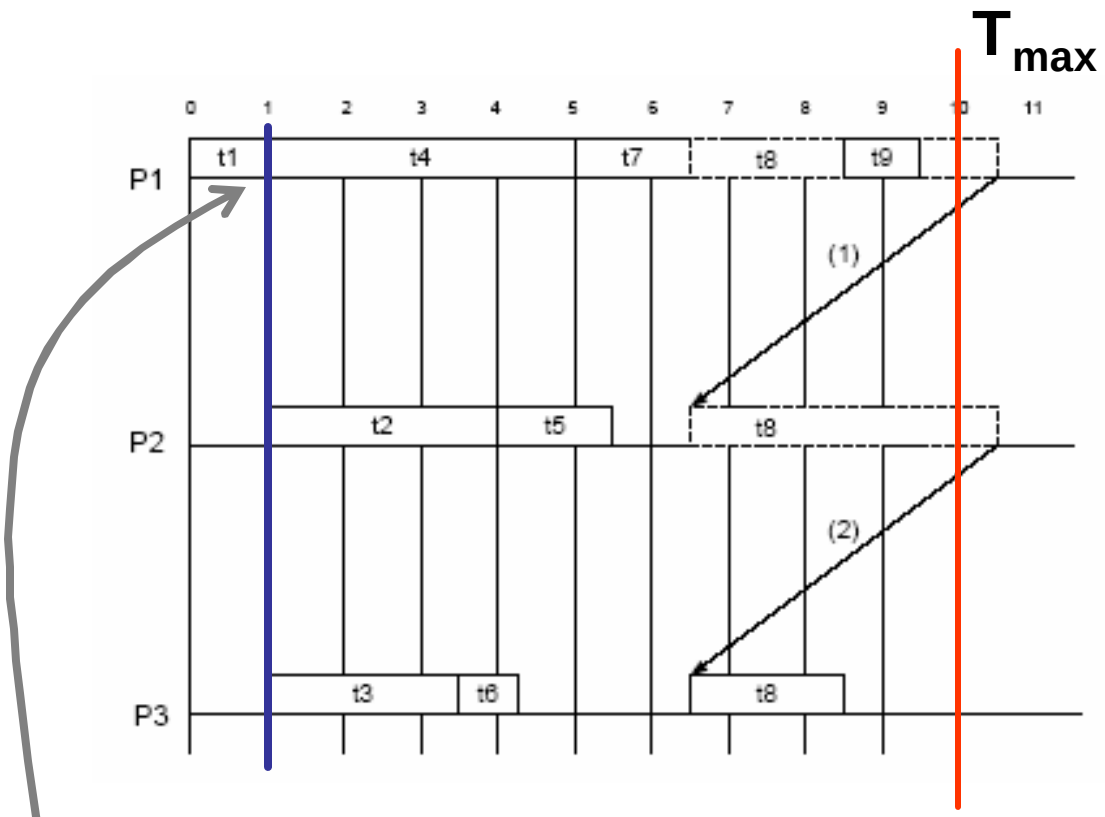
- After the execution of rules 1 or 2,
  - the total cost is updated
  - the completion time of each task allocated is updated
  - If $T > T_{max}$, then the least expensive computing resource is removed from C+ and the algorithm backtracks to the previous allocation instant.
  - The allocation is then attempted again with the reduced C+. If C+ becomes empty the heuristic backtracks to the previous allocation instant.

Heuristic Numerical Example



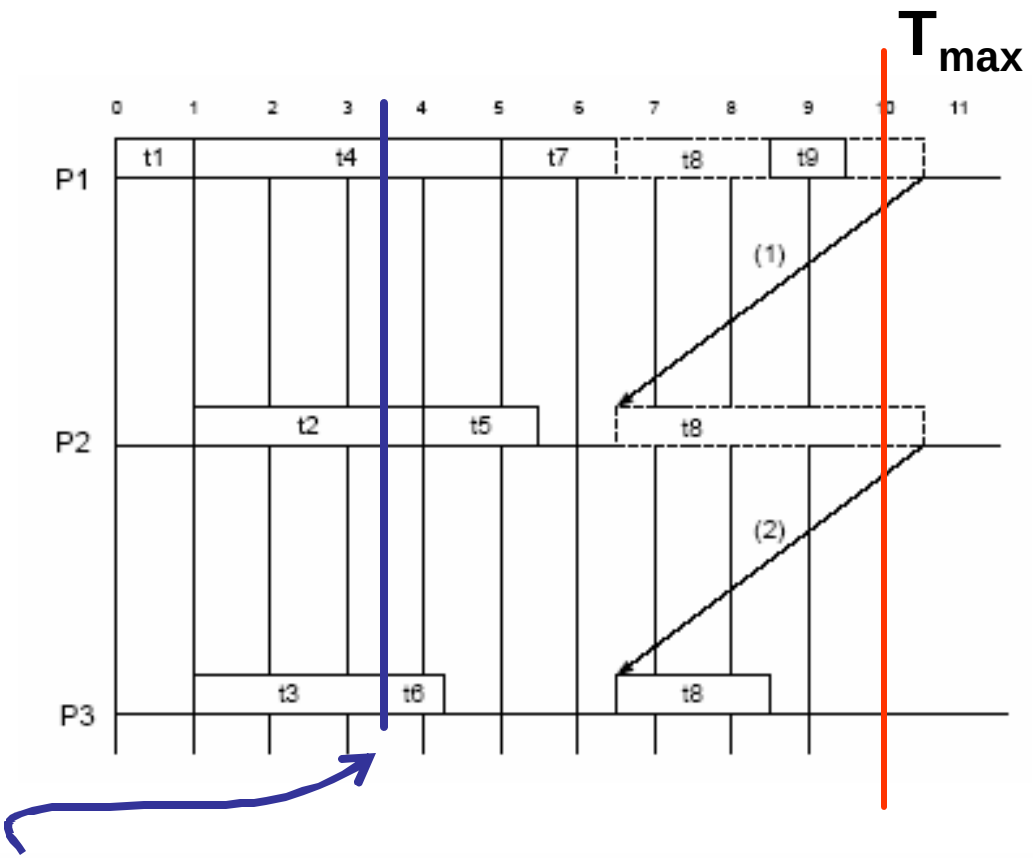
$T = 0$, cost= 0, $\mathcal{T}^+ = \{t_1\}$, $\mathcal{C}^+ = \{p_1, \cdots, p_6\}$:
allocate t_1 to p_1.

Heuristic Numerical Example



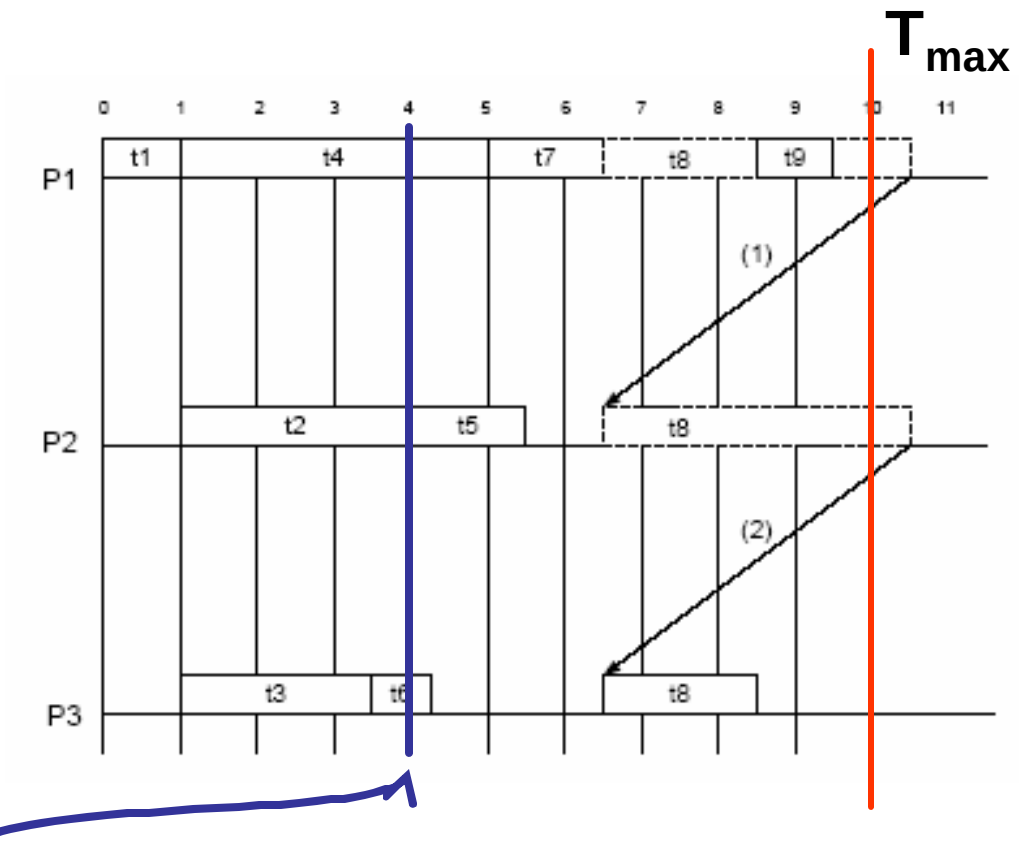
$T = 1$, cost$= 1 \times 0.2 = 0.2$, $\mathcal{T}^+ = \{t_2, t_3, t_4\}$, $\mathcal{C}^+ = \{p_1, \cdots, p_6\}$: allocate t_2 to p_2, t_3 to p_3, and t_4 to p_1.

Heuristic Numerical Example



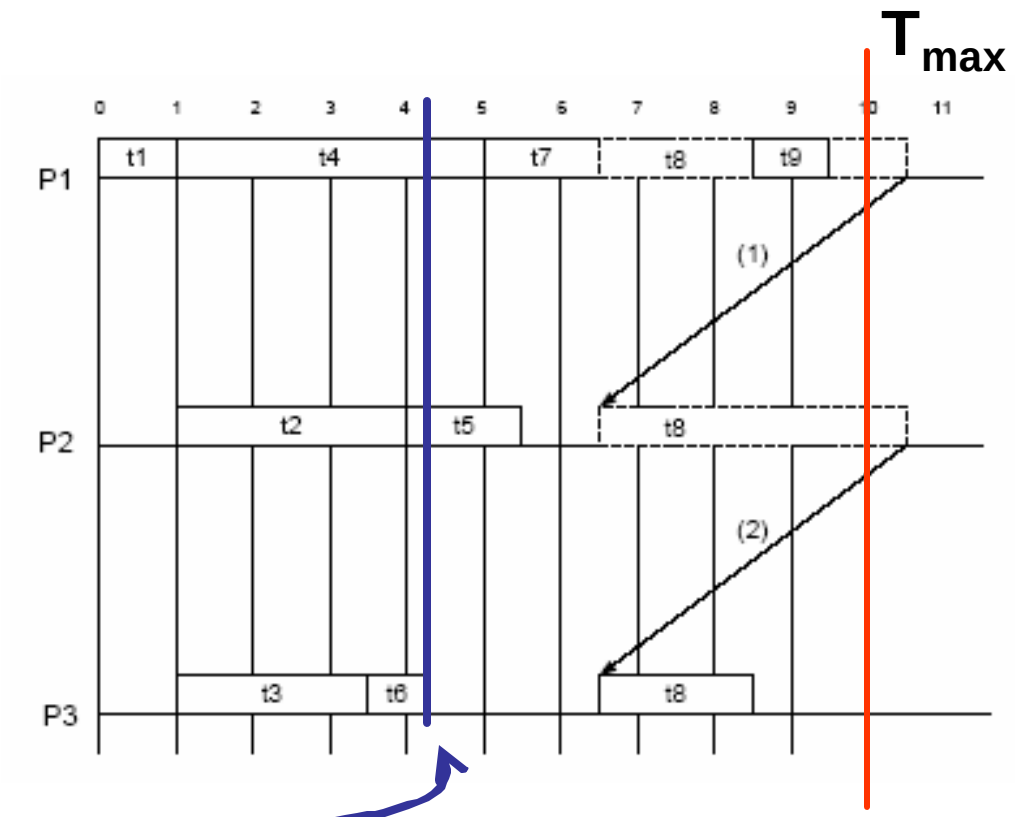
$T = 3.5$, cost$= 0.2 + 2.5 \times 0.35 + 4 \times 0.2 + 3 \times 0.2 = 2.475$, $\mathcal{T}^+ = \{t_6\}$, $\mathcal{C}^+ = \{p_3, \cdots, p_6\}$: allocate t_6 to p_3.

Heuristic Numerical Example



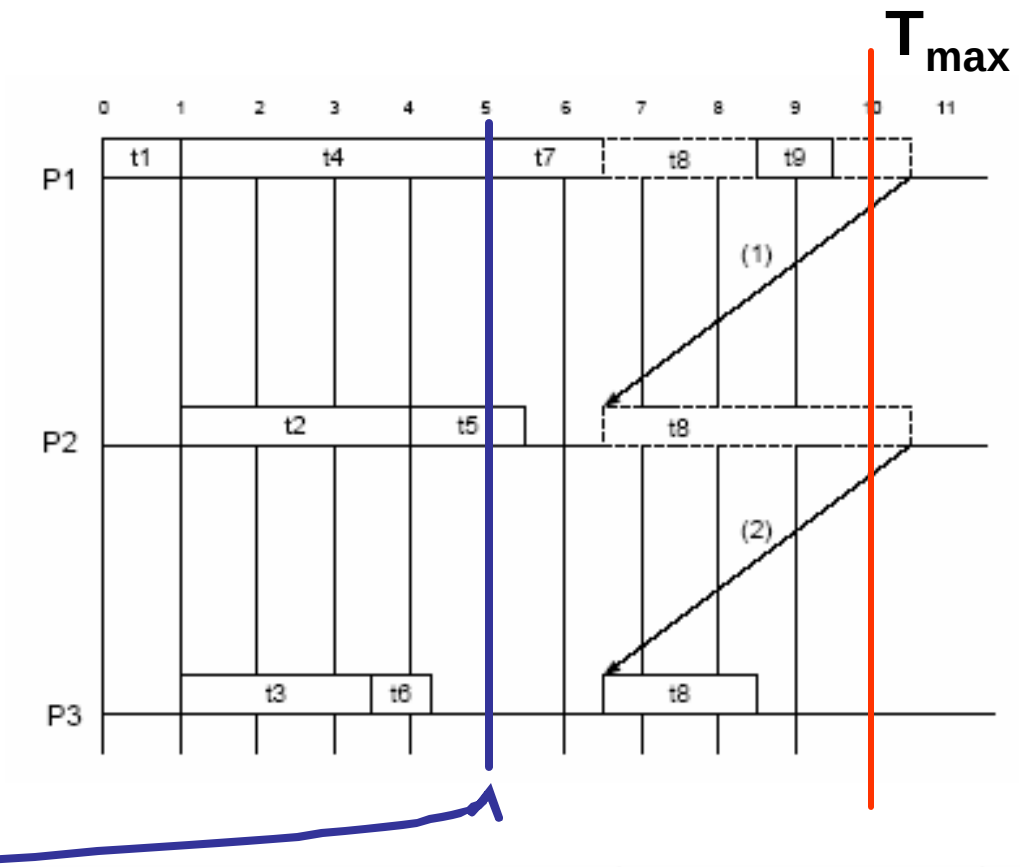
$T = 4$, cost= $2.475 + 0.75 \times 0.35 = 2.7375$, $\mathcal{T}^+ = \{t_5\}$, $\mathcal{C}^+ = \{p_2, p_4, p_5, p_6\}$: allocate t_5 to p_2.

Heuristic Numerical Example



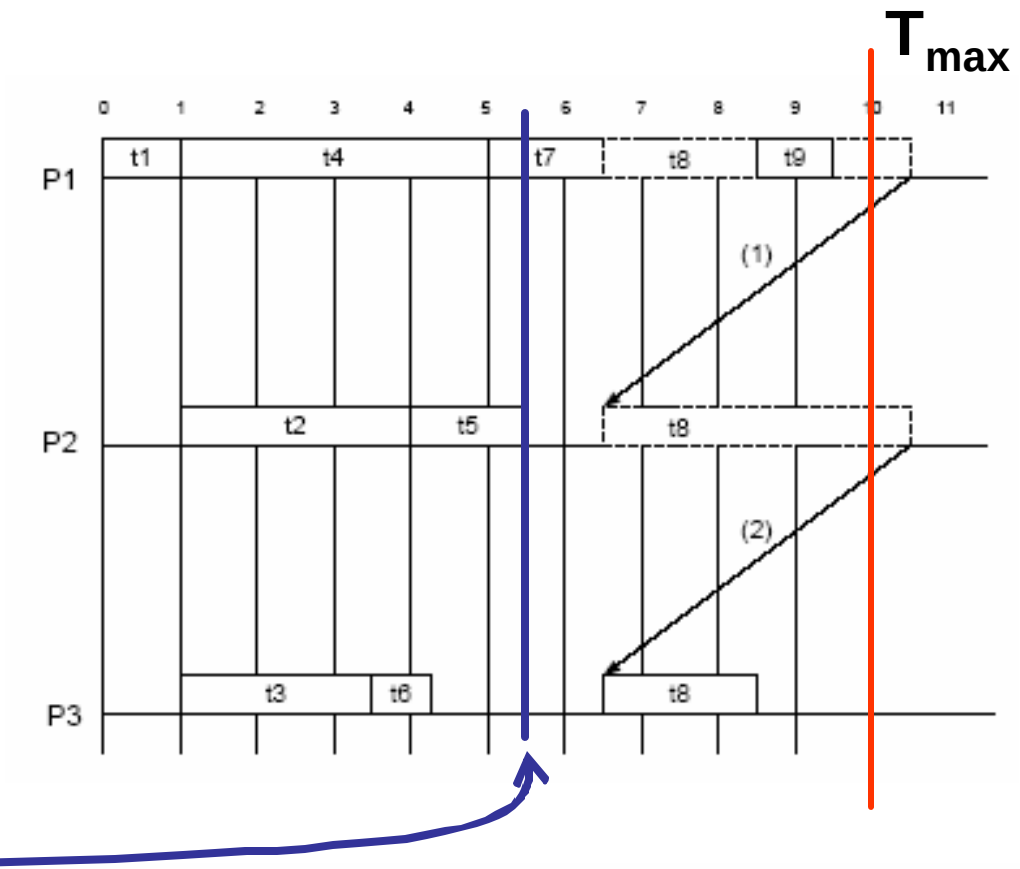
$T = 4.25$, cost= $2.7375 + 1.5 \times 0.2 = 3.0375$, $\mathcal{T}^+ = \{\}$, $\mathcal{C}^+ = \{p_3, p_4, p_5, p_6\}$: no task can be allocated.

Heuristic Numerical Example



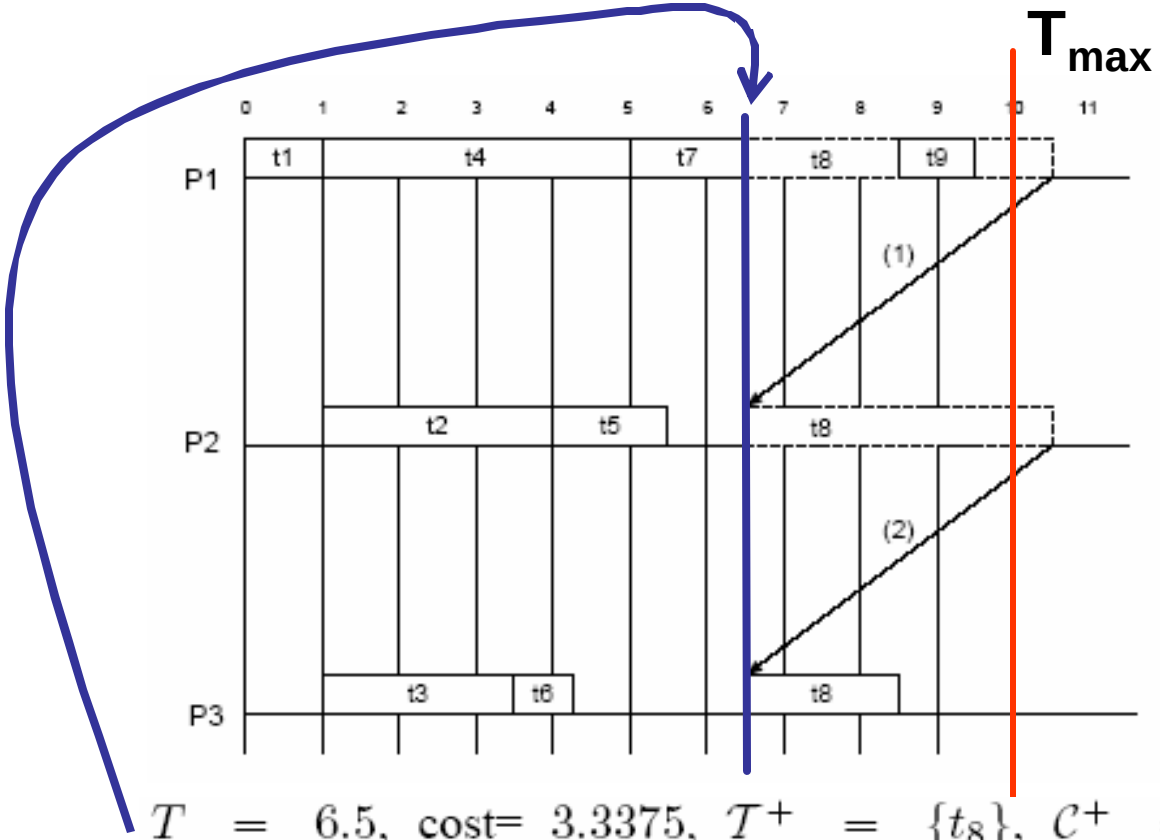
$T = 5$, cost= 3.0375, $\mathcal{T}^+ = \{t_7\}$, $\mathcal{C}^+ = \{p_1, p_3, p_4, p_5, p_6\}$: allocate t_7 to p_1.

Heuristic Numerical Example



$T = 5.5$, cost$= 3.0375 + 1.5 \times 0.2 = 3.3375$, $\mathcal{T}^+ = \{\}$,
$\mathcal{C}^+ = \{p_2, p_3, p_4, p_5, p_6\}$: no task can be allocated.

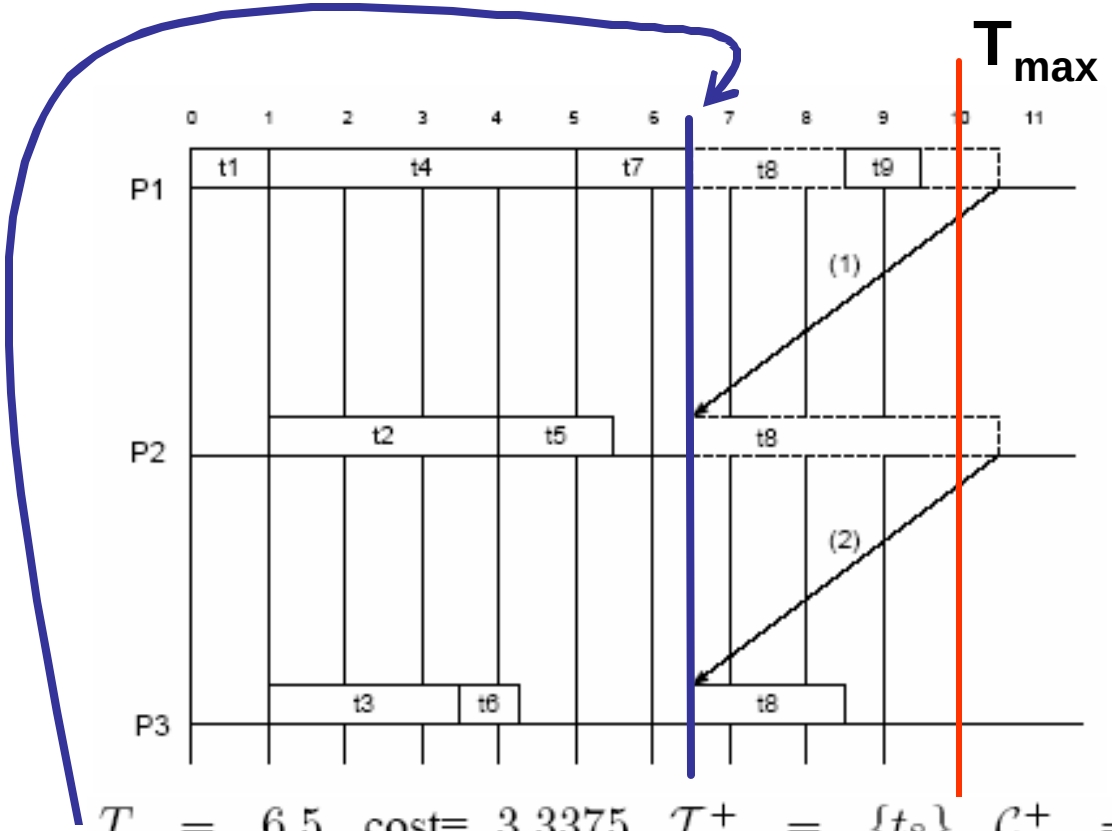
Heuristic Numerical Example



$T = 6.5$, cost= 3.3375, $\mathcal{T}^+ = \{t_8\}$, $\mathcal{C}^+ = \{p_1, \cdots, p_6\}$: allocate t_8 to p_1.

$T = 10.5 > 10$ ($T_{\max}$). Backtrack to previous allocation instant (see arrow labeled (1) in Fig. 5) and remove p_1 from $\mathcal{C}^+$.

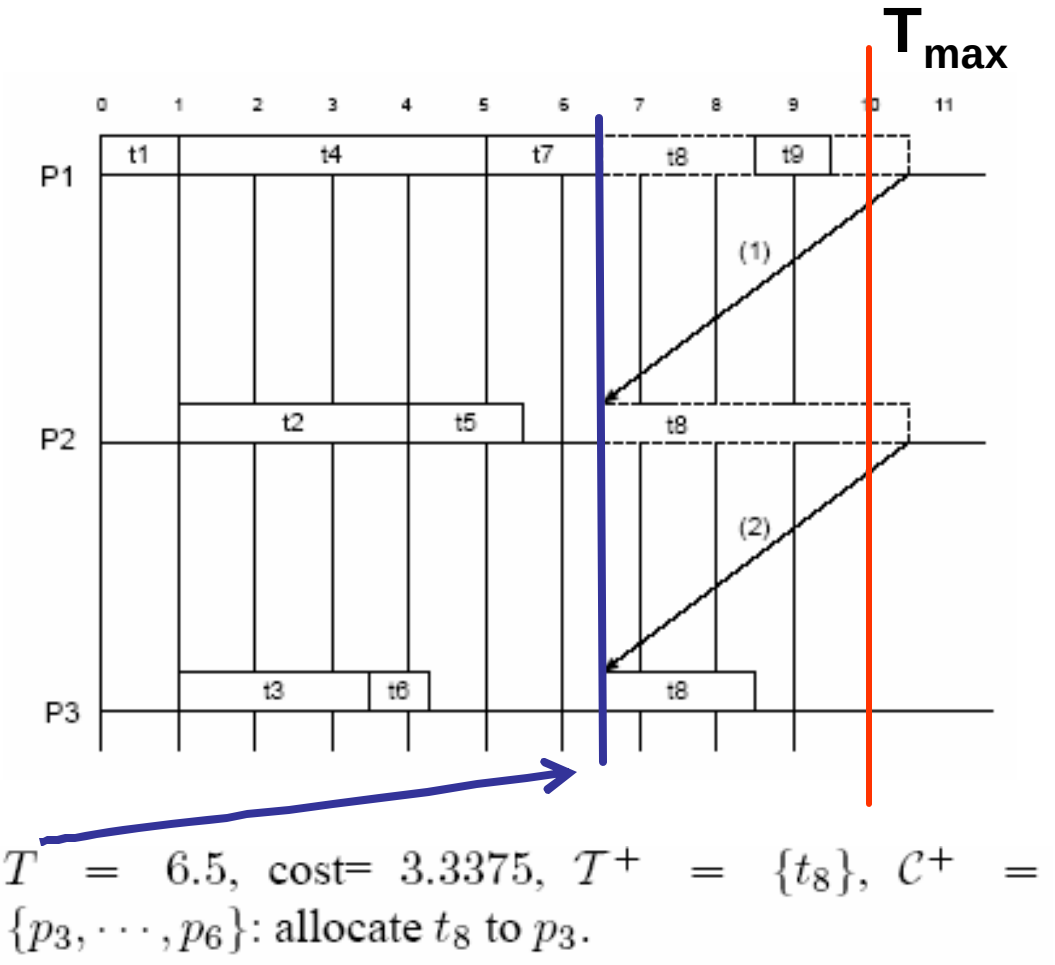
Heuristic Numerical Example



$T = 6.5$, cost= 3.3375, $\mathcal{T}^+ = \{t_8\}$, $\mathcal{C}^+ = \{p_2, \cdots, p_6\}$: allocate t_8 to p_2.

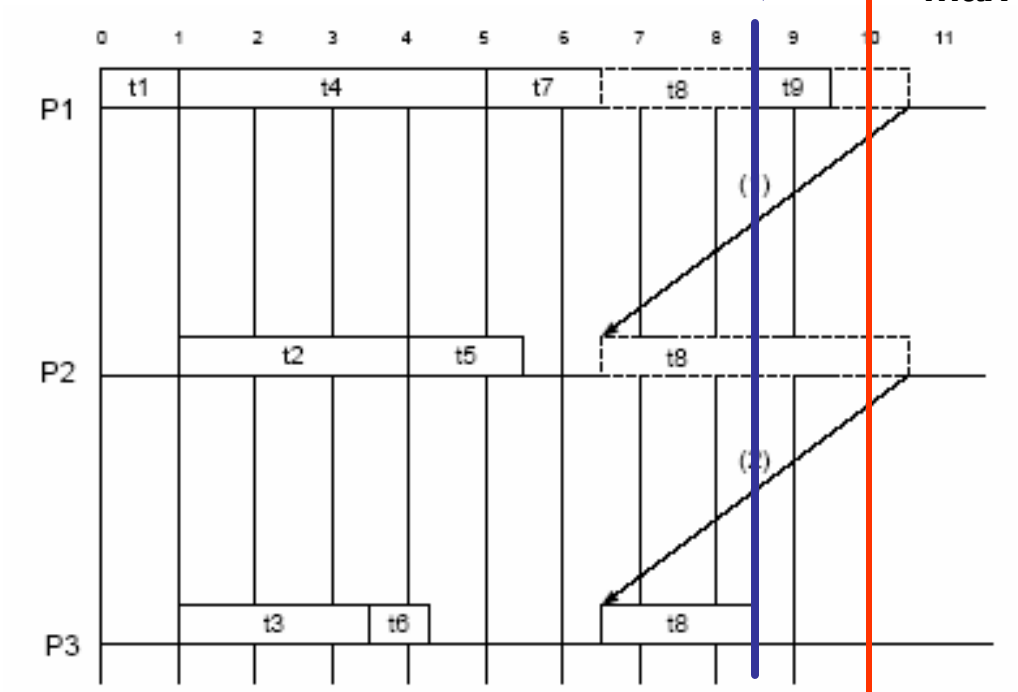
$T = 10.5 > 10$ ($T_{\max}$). Backtrack to previous allocation instant (see arrow labeled (2) in Fig. 5) and remove p_2 from $\mathcal{C}^+$.

Heuristic Numerical Example



$T = 6.5$, cost= 3.3375, $\mathcal{T}^+ = \{t_8\}$, $\mathcal{C}^+ = \{p_3, \cdots, p_6\}$: allocate t_8 to p_3.

Heuristic Numerical Example



$T < T_{max}$

$T = 8.5$, cost $= 3.3375 + 2 \times 0.35 = 4.0375$, $\mathcal{T}^+ = \{t_9\}$, $\mathcal{C}^+ = \{p_1, \cdots, p_6\}$: allocate t_9 to p_1.

$T = 9.5$, cost $= 4.0375 + 1 \times 0.2 = 4.2375$, $\mathcal{T}^+ = \{\}$, $\mathcal{C}^+ = \{p_1, \cdots, p_6\}$: all tasks have been allocated. Stop.

Outline

- Introduction to Grid Computing
- A motivating Example
- QoS Metrics in Grid Computing
- A simple example of the resource allocation problem
- Formalization of the resource allocation problem
- An example of a heuristic
- Concluding remarks

Concluding Remarks

- Grid architectures provide an interesting service-oriented platform for both scientific and commercial applications.
- Still open problems: capacity planning, performance modeling and analysis.
- Tough problems still requiring more work: resource scheduling and allocation.

Additional Bibliography

- "A Framework for Resource Allocation in Grid Computing," D. Menascé and E. Casalicchio, *Proc. 12th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS),* Volendam, The Netherlands, October 5-7, 2004.
- "Mapping Service Level Agreements in Distributed Applications," D. Menascé, *IEEE Internet Computing*, Vol.8, No. 5, September/October 2004.
- "QoS in Grid Computing," D. Menascé and E. Casalicchio, *IEEE Internet Computing*, July/August 2004, Vol. 8, No. 4.