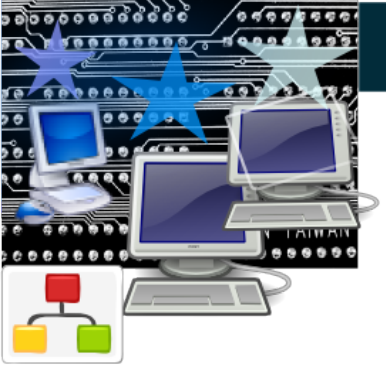


Towards PaaS and Clouds Our experience with BonjourGrid and PastryGrid – AOC Team –

Christophe Cérin[1]

[1]Université de Paris XIII, CNRS UMR 7030, France

Bi-lateral China-France workshop

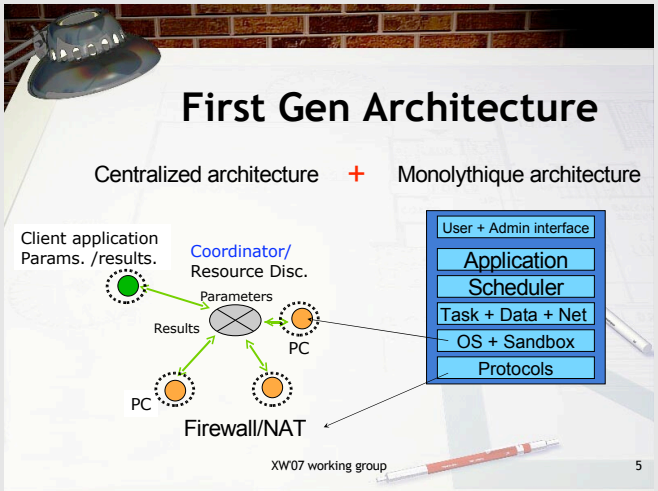
Table of contents

Objectives

1. Motivate research projets in Grids & Clouds ;
2. Starting from recent advances in Desktop Grid Middleware:
   - BonjourGrid (orchestration of multiple instances of DG middleware) and PastryGrid (fully distributed execution of applications)
   - Joint works with UTIC lab., Tunisia (Heithem Abbes and Mohamed Jemni)
3. Before keeping innovative ideas to reuse in Cloud Architectures / Systems:
   - decentralized architectures and services;
   - large scale systems (FT);
   - interoperability of services (the client is not a prisonner, or if it is, he can choose his prison(s)!)

Desktop Grid Architectures

Desktop Grid

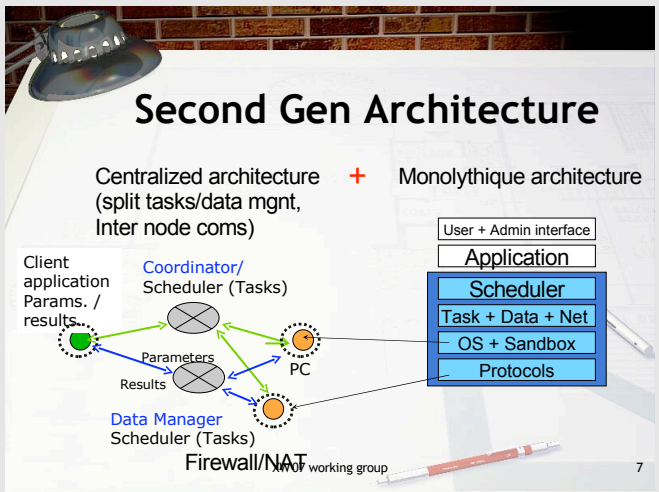


Key Points

- Federation of thousand of nodes;
- Internet as the communication layer: no trust!
- Volatility; local IP; Firewall

Desktop Grid Architectures

Desktop Grid



Future Generation (in 2006)

- Distributed Architecture
- Architecture with modularity: every component is "configurable": scheduler, storage, transport protocole
- Direct communications between peers;
- Security;
- Applications coming from any sciences (e-Science applications)

In search of distributed architecture

First line: publish/subscribe system to notify and coordinate services and multiple DG without a central broker ⇒ BonjourGrid;

Second line: approach based on structured overlay network to discover (on the fly) the next node executing the next task ⇒ PastryGrid;

(main contributions of Heithem Abbes in his PhD)

Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);

Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;

Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;
- Reduce as much as possible the use of any central element;

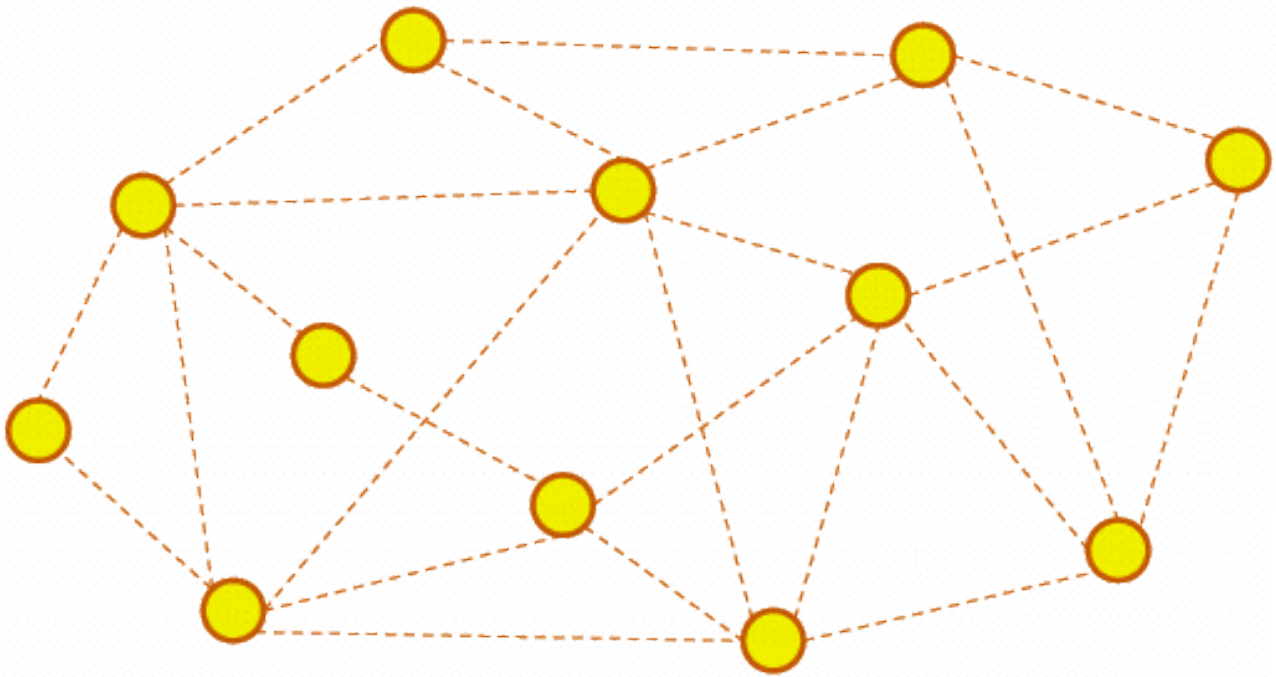
Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;
- Reduce as much as possible the use of any central element;
- Create a coordinator, on the fly, without any system administrator intervention; From a vision with a single coordinator towards a vision with multiple coordinators.

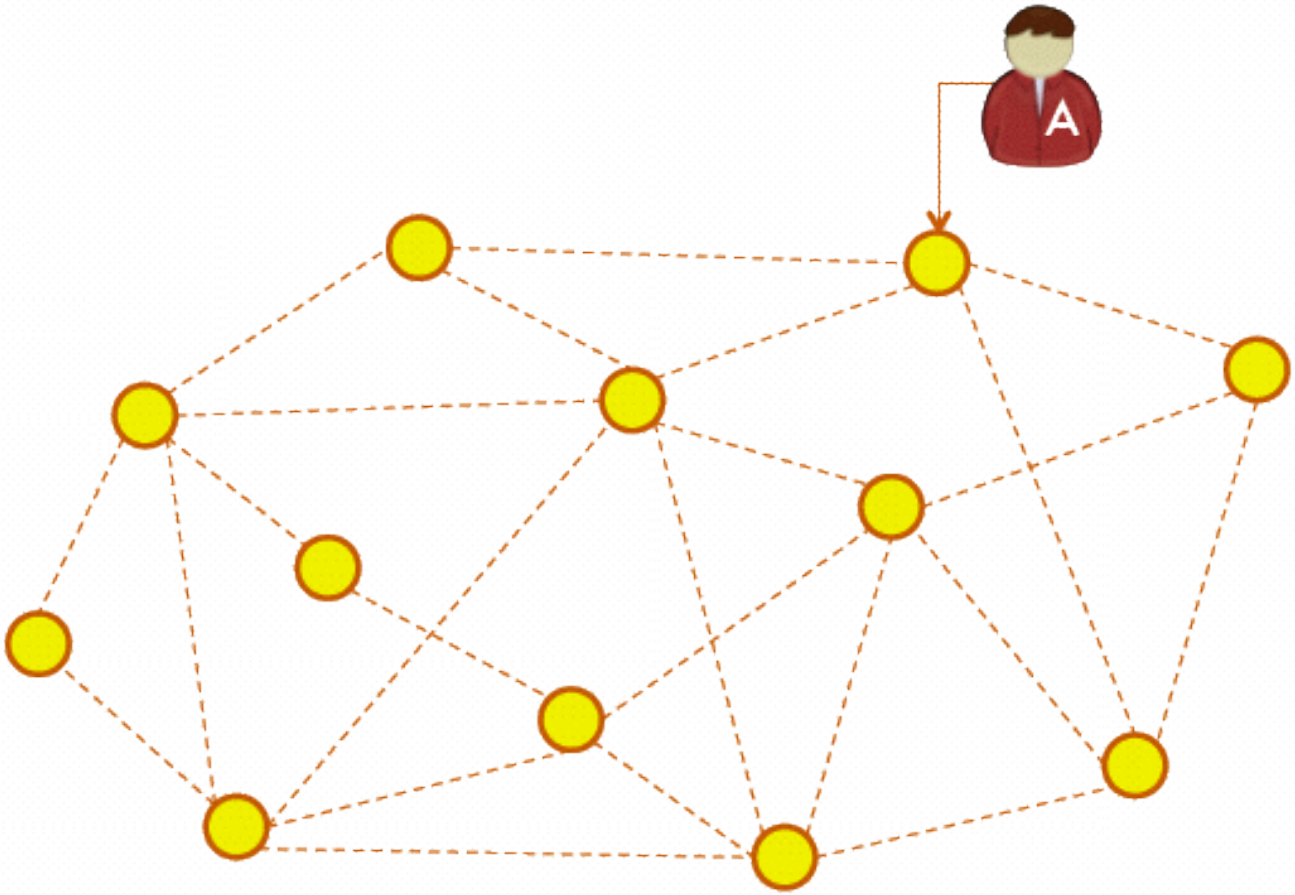
Main objectives of BonjourGrid

- Count on existing distributed tools for services discovering (publish/subscribe paradigm);
- Design and implement a platform able to manage multiple instances of DG middleware;
- Reduce as much as possible the use of any central element;
- Create a coordinator, on the fly, without any system administrator intervention; From a vision with a single coordinator towards a vision with multiple coordinators.
- Each coordinator searches, in a concurrent way, participants (idle machines)

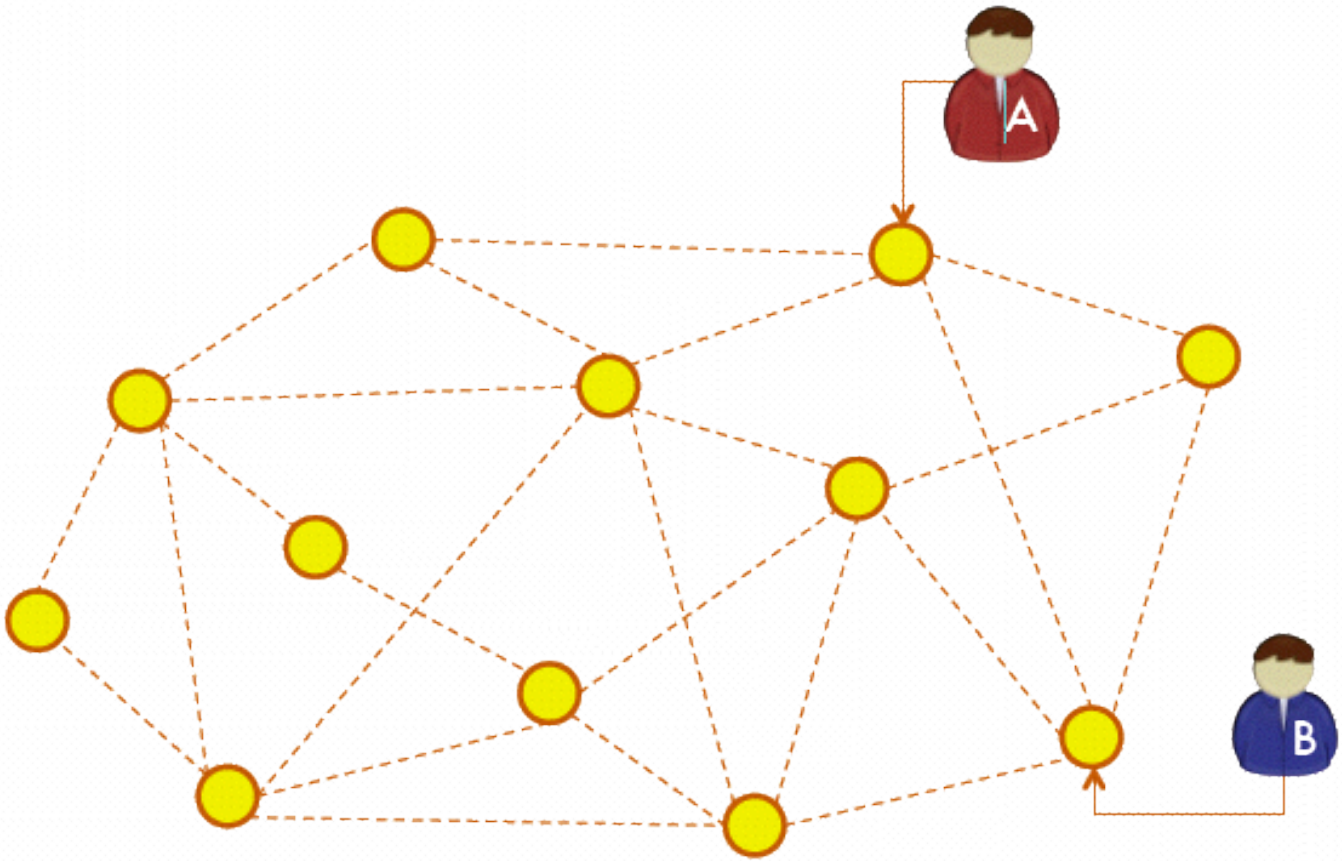
How BonjourGrid works



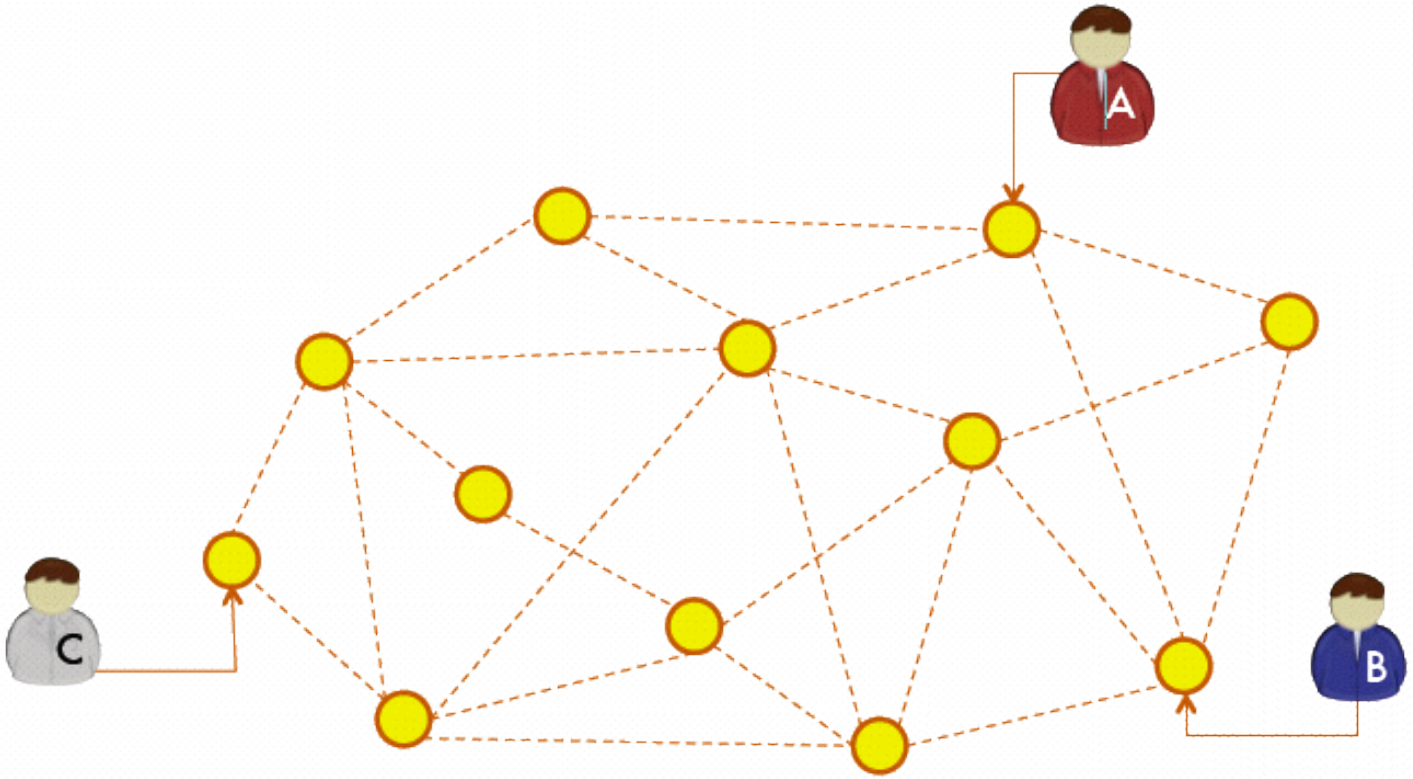
How BonjourGrid works



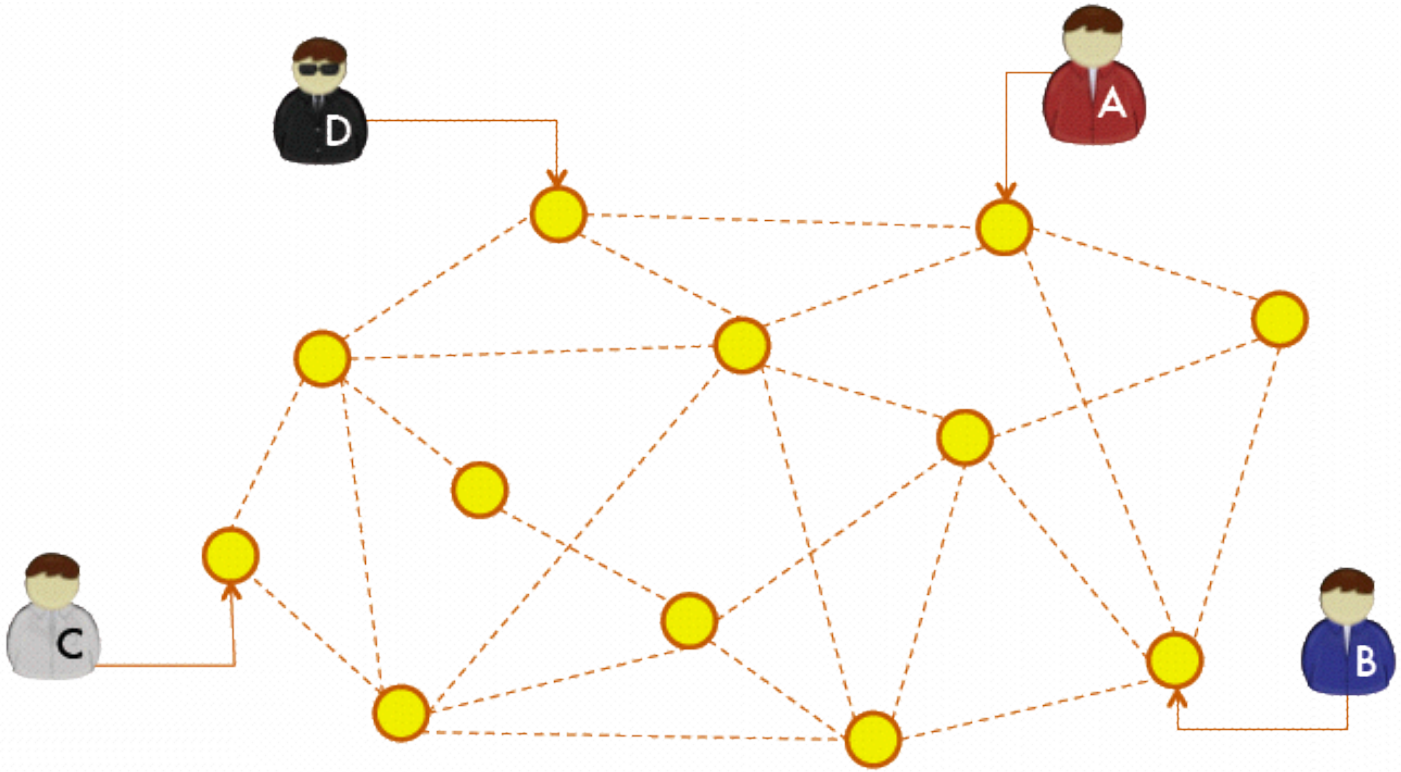
How BonjourGrid works



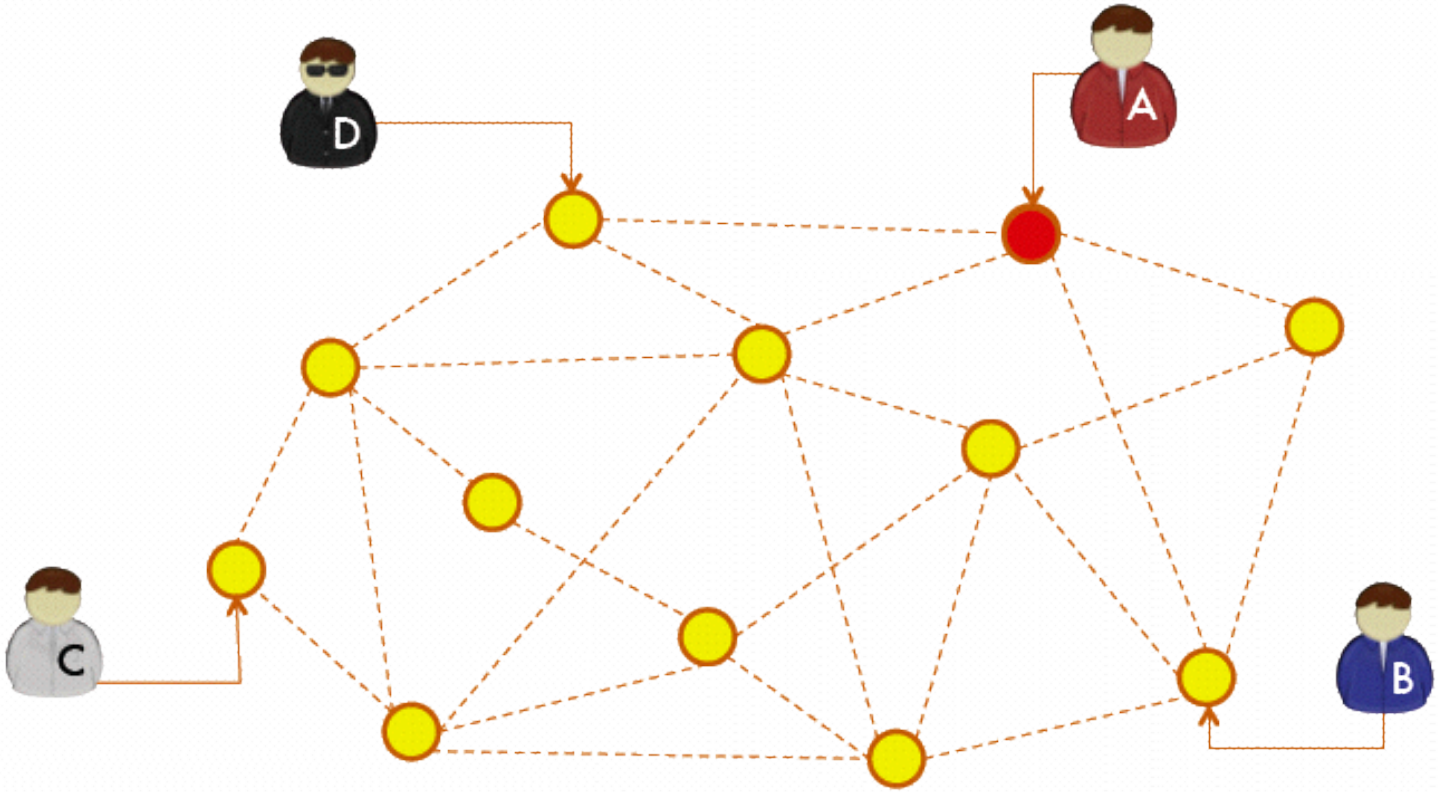
How BonjourGrid works



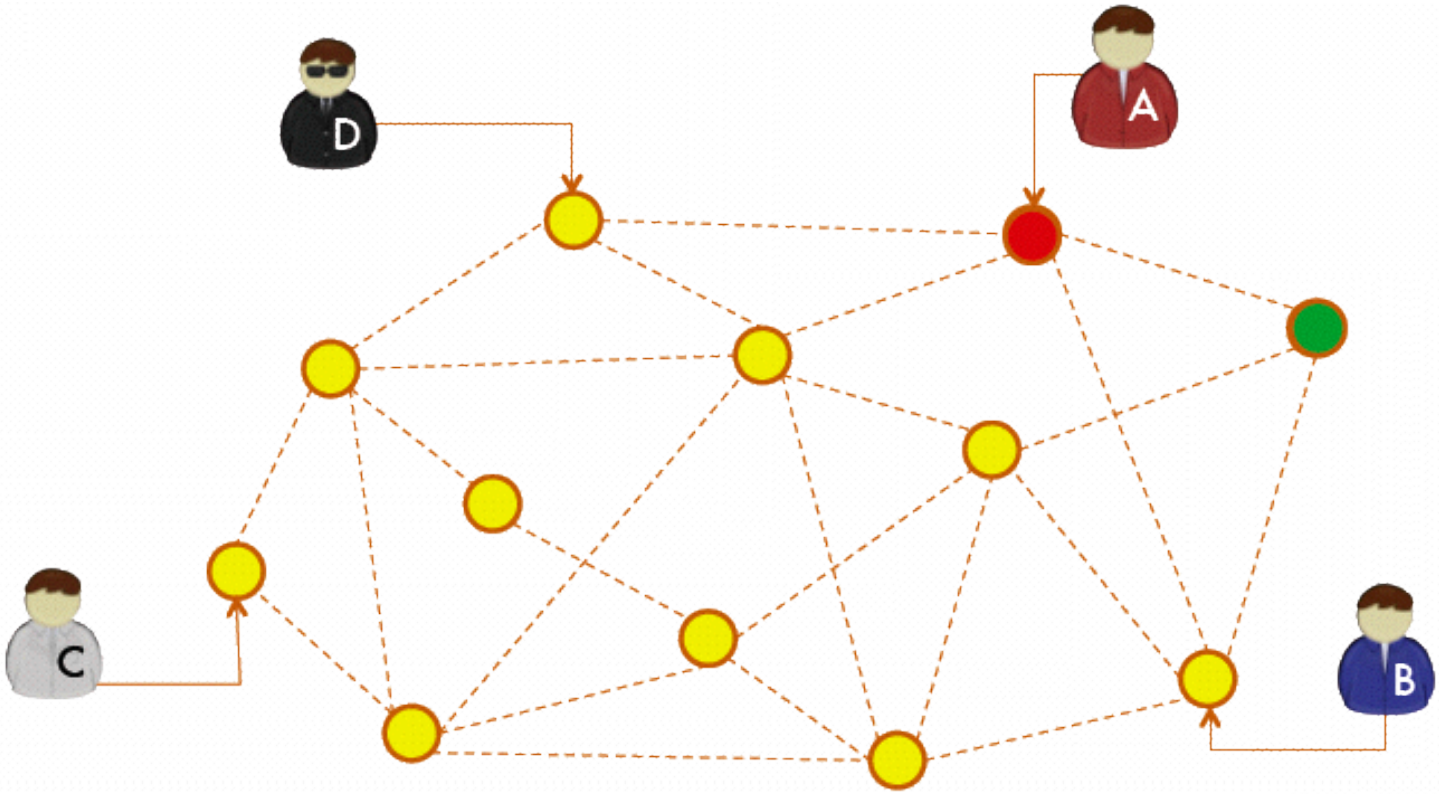
How BonjourGrid works



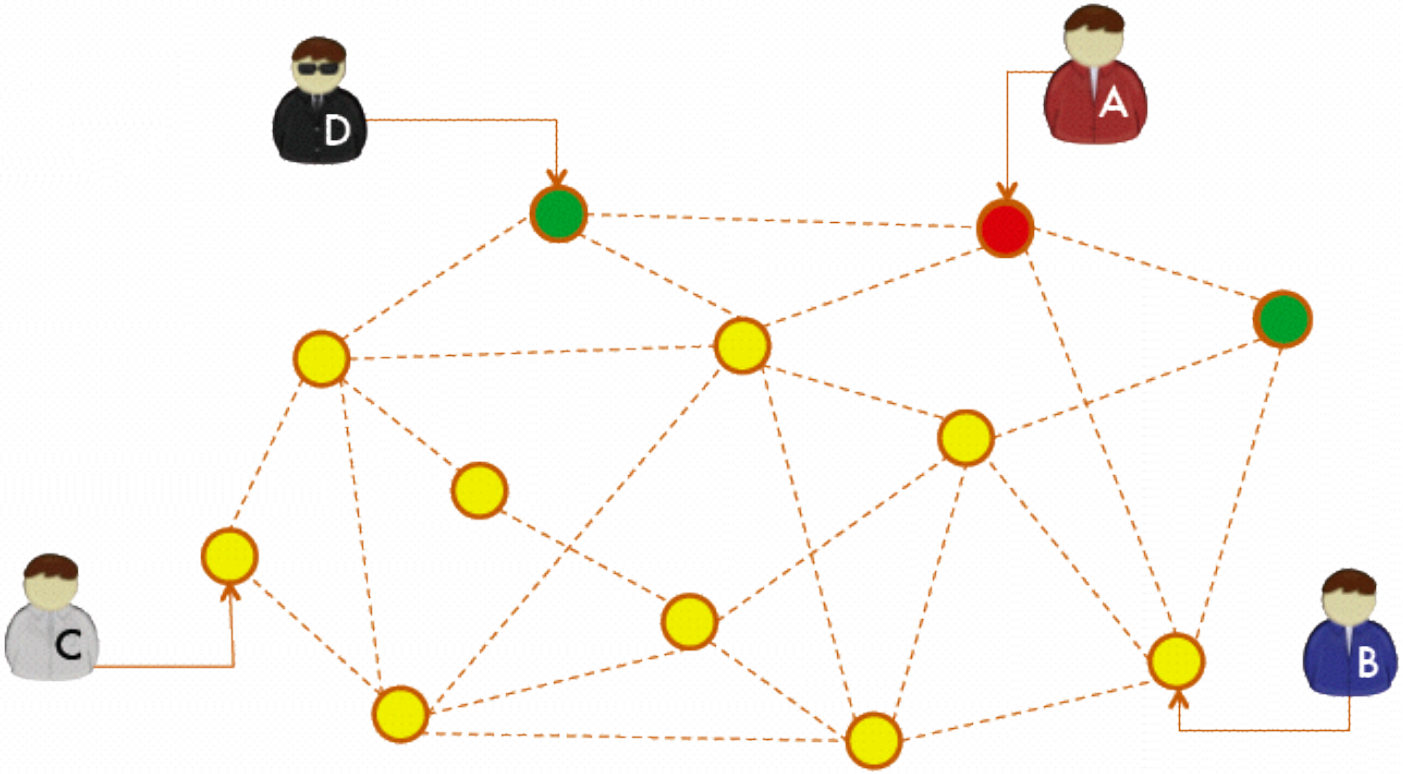
How BonjourGrid works



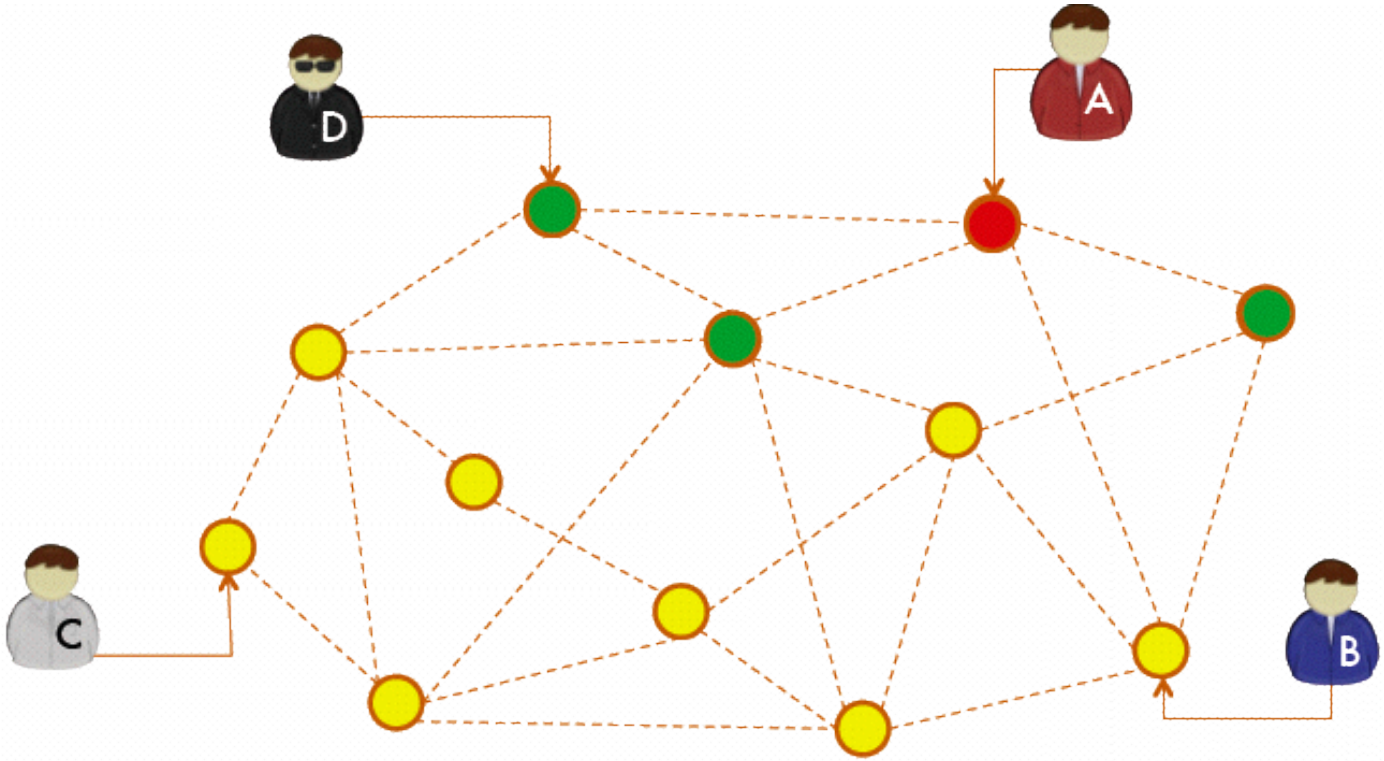
How BonjourGrid works



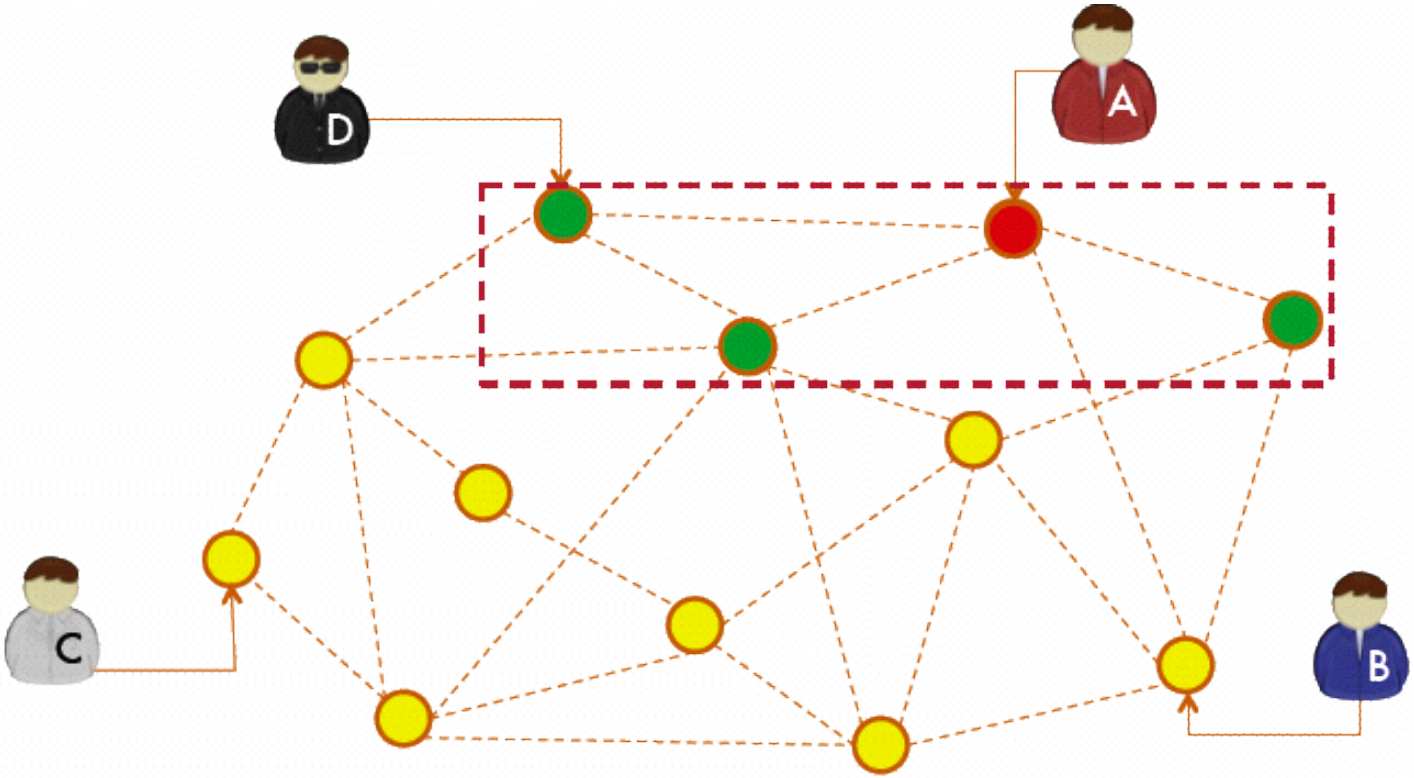
How BonjourGrid works



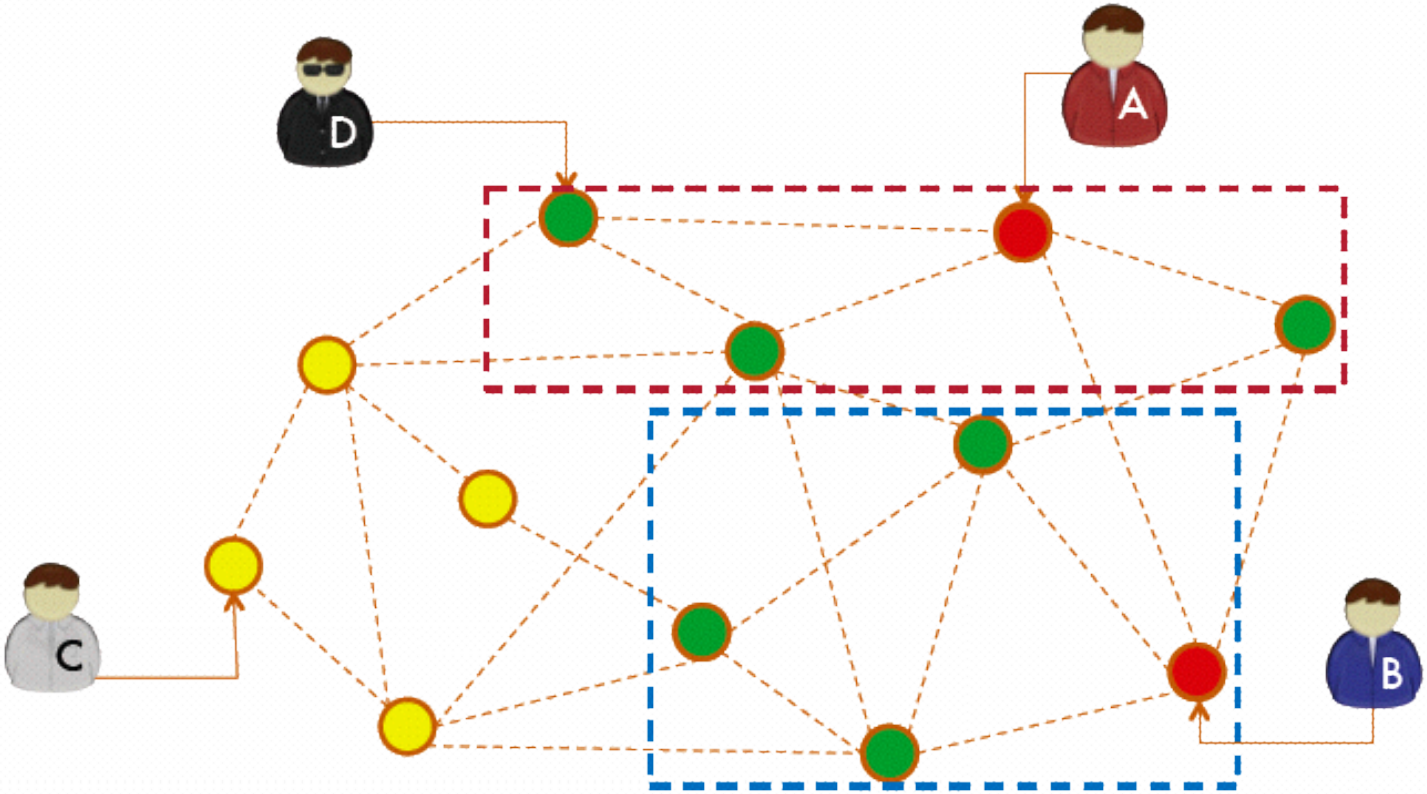
How BonjourGrid works



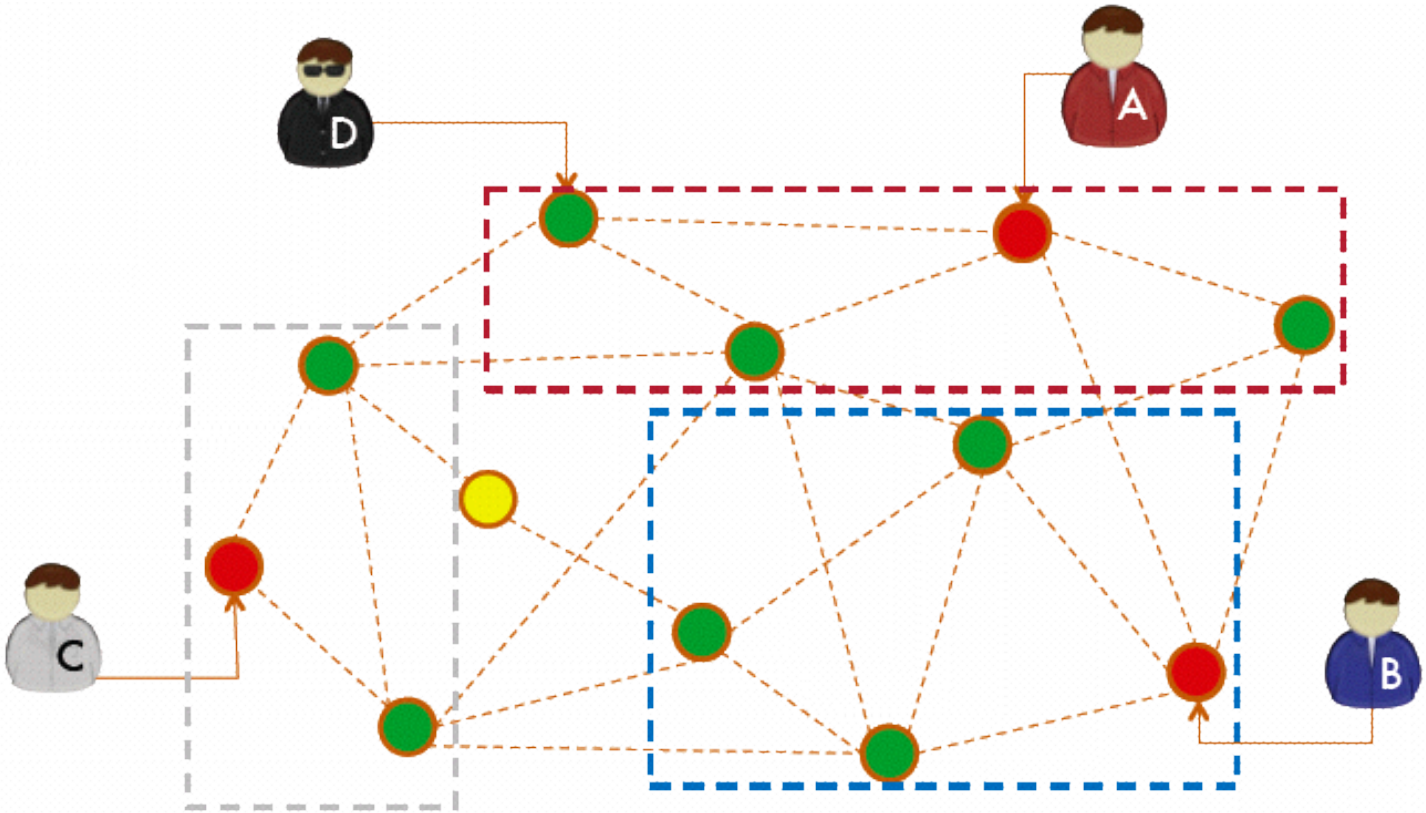
How BonjourGrid works



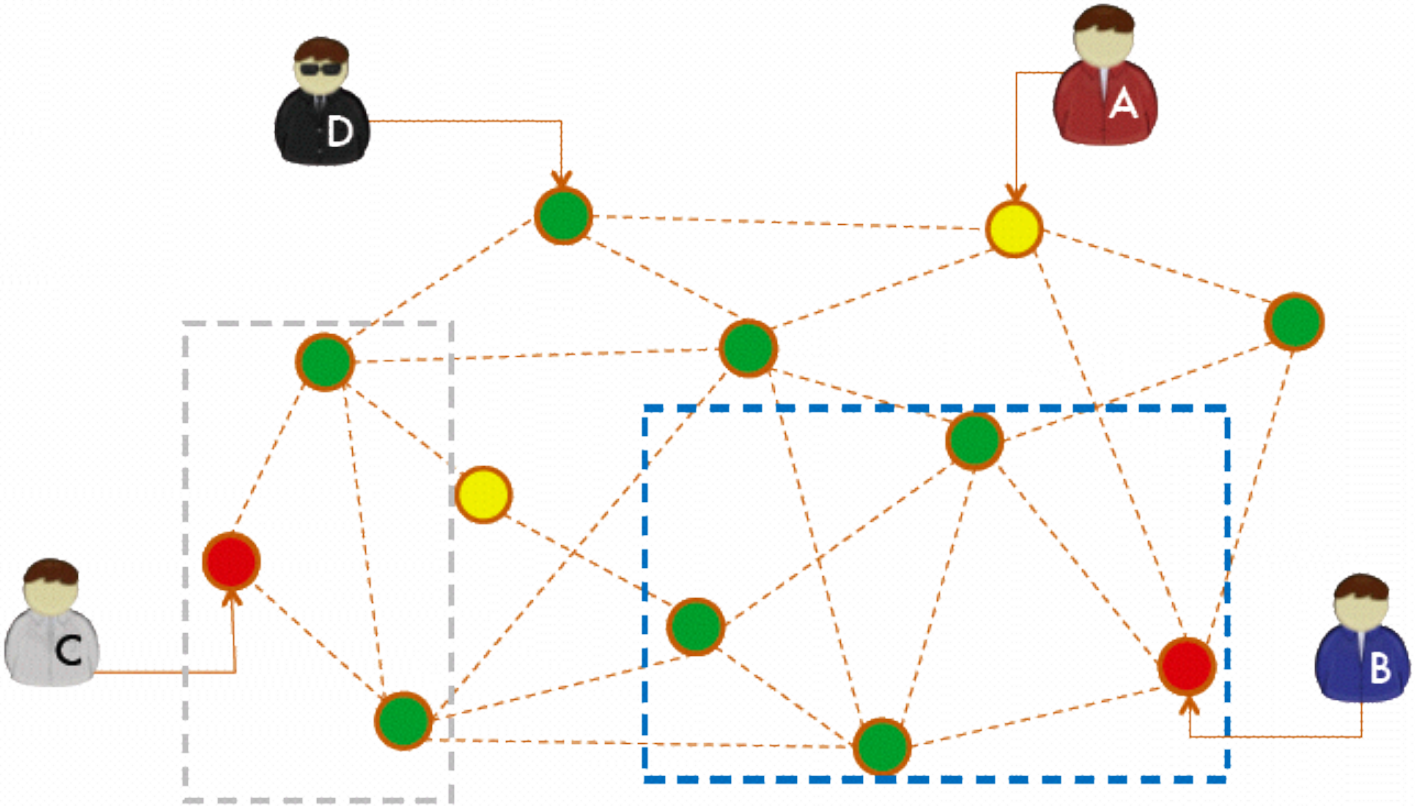
How BonjourGrid works



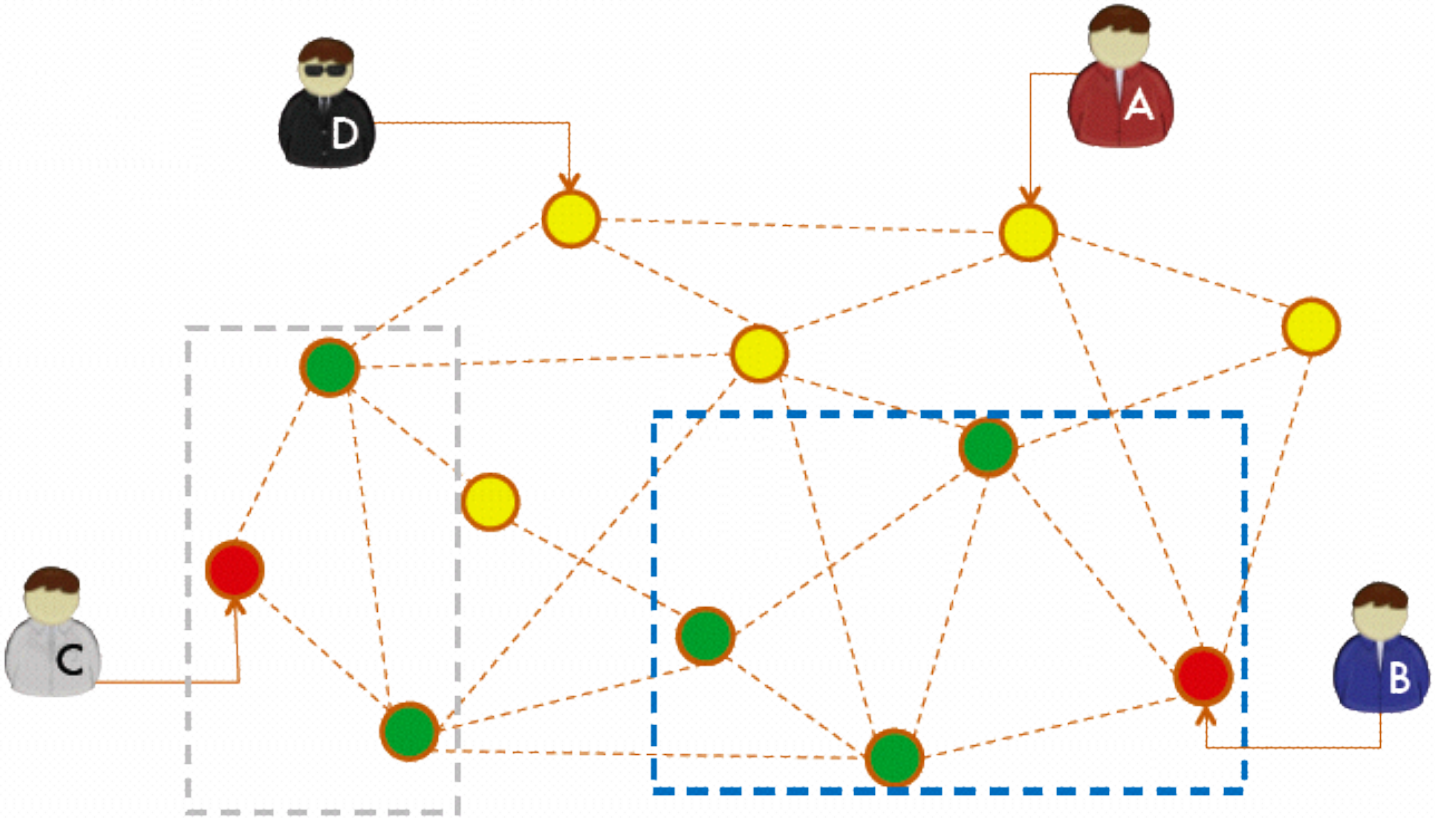
How BonjourGrid works



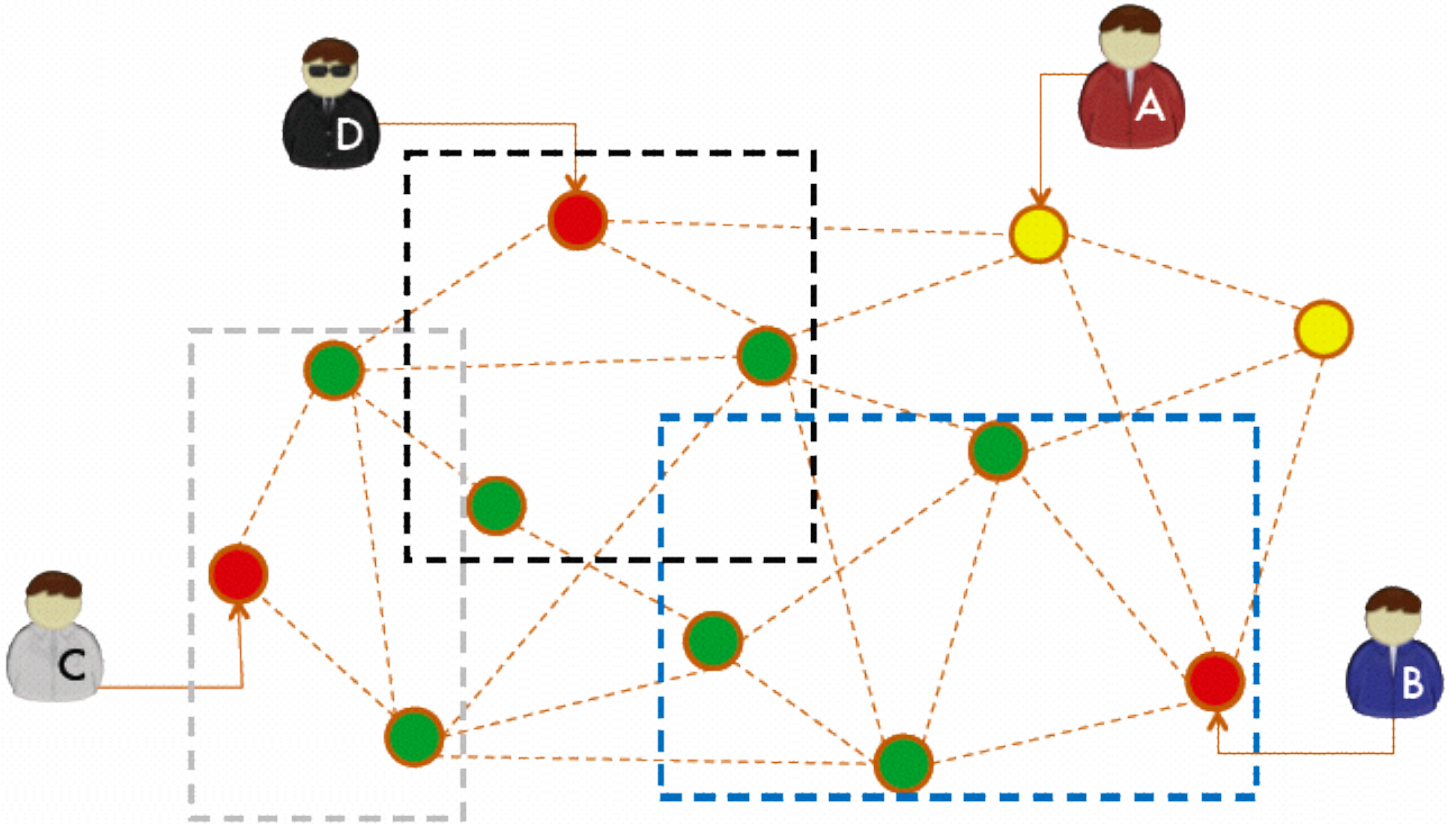
How BonjourGrid works



How BonjourGrid works



How BonjourGrid works



BonjourGrid vision

- The user requests for computation;

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs...)

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs...)
- Upon completion, the coordinator returns to the idle state, slaves are freed and the coordination protocol:

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs...)
- Upon completion, the coordinator returns to the idle state, slaves are freed and the coordination protocol:
  - manages and controls resources, services and computing elements;
  - does not depend on any specific machine nor any central element.

BonjourGrid vision

- The user requests for computation;
- The user provides the control flow graph, binaries, input data;
- The user deploys locally a coordinator and requests for participants; We support XtremWeb, Condor, Boinc.
- The coordinator selects a set of machines (criteria: RAM, CPU, costs...)
- Upon completion, the coordinator returns to the idle state, slaves are freed and the coordination protocol:
  - manages and controls resources, services and computing elements;
  - does not depend on any specific machine nor any central element.

The protocol for resources discovering

- Based on Bonjour from Apple;

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)
- Concerning the Wide Area implementation: we can also think about Apache Kandula (http://ws.apache.org/kandula/) or even cisco Jabber protocol (http://www.jabber.com):

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)
- Concerning the Wide Area implementation: we can also think about Apache Kandula (http://ws.apache.org/kandula/) or even cisco Jabber protocol (http://www.jabber.com):
  - Extensible Messaging and Presence Protocol (XMPP) (formerly named Jabber) is an open, XML-based protocol originally aimed at near-real-time, extensible instant messaging (IM) and presence information, but now expanded into the broader realm of message-oriented middleware

The protocol for resources discovering

- Based on Bonjour from Apple;
- A publish/subscribe system is easy to use (toolbox = publish(), subscribe(), browse())
- Bonjour is dedicated to LAN (wide area Bonjour? We need some experiments)
- Concerning the Wide Area implementation: we can also think about Apache Kandula (http://ws.apache.org/kandula/) or even cisco Jabber protocol (http://www.jabber.com):
  - Extensible Messaging and Presence Protocol (XMPP) (formerly named Jabber) is an open, XML-based protocol originally aimed at near-real-time, extensible instant messaging (IM) and presence information, but now expanded into the broader realm of message-oriented middleware
- The current protocol has been developed/specified with 'ad-hoc' methods → we need to consolidate the trust (ongoing project to verify it, based on Colored Petri Nets)

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;
- We assume that any coordinator is responsible for its FT (it manages the volatility of attached slaves)

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;
- We assume that any coordinator is responsible for its FT (it manages the volatility of attached slaves)
- Our solution: tolerate the failure of coordinators

Fault Tolerance with BonjourGrid

- Intrinsic property of any large scale system;
- We assume that any coordinator is responsible for its FT (it manages the volatility of attached slaves)
- Our solution: tolerate the failure of coordinators
  - For any application we create and manage dynamically copies of the coordinator;
  - We manage k copies; based on passive replication.
  - When a service disappears: we added a special status flag to distinguish between 'end of the application' / 'failure' $\Rightarrow$ slaves can redirect the communication to a copy.

Intensive Experiments

- BonjourGrid has been tested intensively: stressed scenario to more relaxing scenario

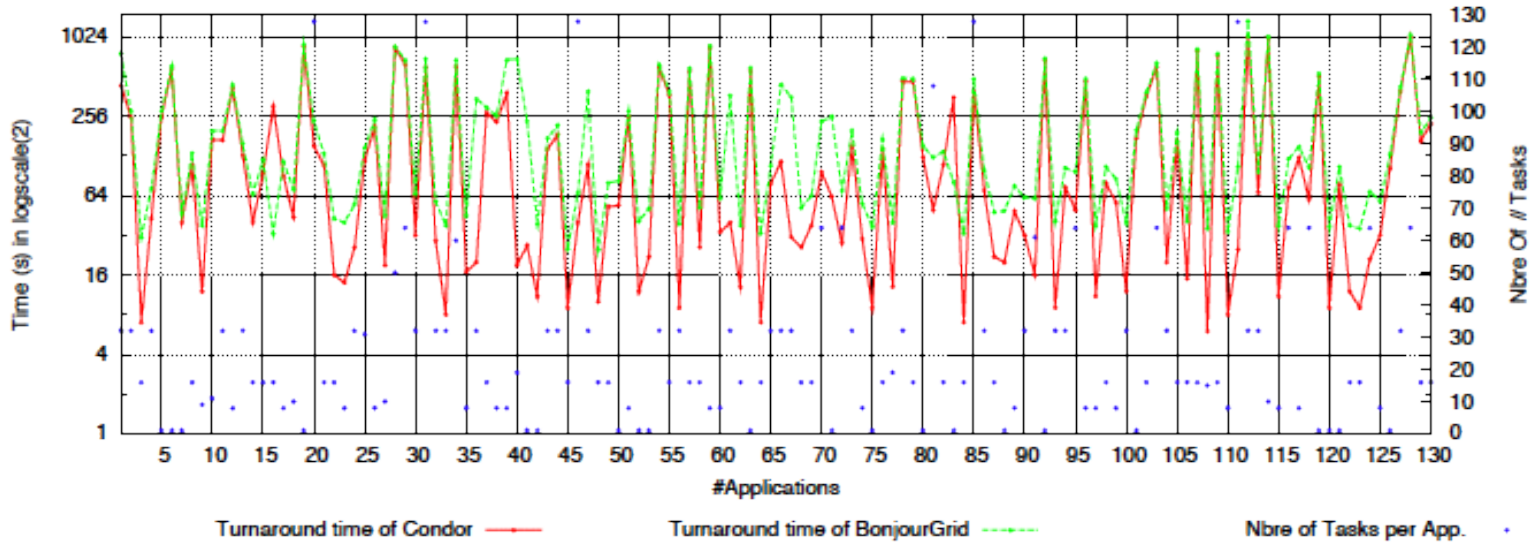
Intensive Experiments

- BonjourGrid has been tested intensively: stressed scenario to more relaxing scenario
  - in terms of #coordinator versus #nodes
  - in terms of using virtual machines to reach 1000 nodes;
  - in terms of comparing Boinc, Condor, XtremWeb over our protocol;
  - in terms of robustness in supporting FT;

Intensive Experiments

- BonjourGrid has been tested intensively: stressed scenario to more relaxing scenario
  - in terms of #coordinator versus #nodes
  - in terms of using virtual machines to reach 1000 nodes;
  - in terms of comparing Boinc, Condor, XtremWeb over our protocol;
  - in terms of robustness in supporting FT;
- Example Condor: 130 applications (2 to 128 // tasks), 200 nodes, application task: 1s to 500s. Result: with BonjourGrid, 35% of applications generate a delay of about 30s.

Experiments: one example



PastryGrid's overview

Main objectives

- Fully distributed execution of task graph;

PastryGrid's overview

Main objectives

- Fully distributed execution of task graph;
- Distributed resource management;

PastryGrid's overview

Main objectives

- Fully distributed execution of task graph;
- Distributed resource management;
- Distributed coordination;

PastryGrid's overview

Main objectives

- Fully distributed execution of task graph;
- Distributed resource management;
- Distributed coordination;
- Dynamically creation of an execution environment;

PastryGrid's overview

Main objectives

- Fully distributed execution of task graph;
- Distributed resource management;
- Distributed coordination;
- Dynamically creation of an execution environment;
- No central element;

PastryGrid's overview

Main objectives

- Fully distributed execution of task graph;
- Distributed resource management;
- Distributed coordination;
- Dynamically creation of an execution environment;
- No central element;

PastryGrid vs BonjourGrid

- New computing platform vs multiple instances of DG middleware

PastryGrid's overview

Main objectives

- Fully distributed execution of task graph;
- Distributed resource management;
- Distributed coordination;
- Dynamically creation of an execution environment;
- No central element;

PastryGrid vs BonjourGrid

- New computing platform vs multiple instances of DG middleware
- Contains its own approach for application distribution vs you count on a DG middleware

PastryGrid's Terminology

Task terminology

- Friend tasks: T_2, T_3 share the same successor (T_6)

PastryGrid's Terminology

Task terminology

- Friend tasks: T_2, T_3 share the same successor (T_6)
- Shared tasks T_6: has $n > 1$ ancestors (T_2, T_3)

PastryGrid's Terminology

Task terminology

- Friend tasks: T_2, T_3 share the same successor (T_6)
- Shared tasks T_6: has $n > 1$ ancestors (T_2, T_3)
- Isolated tasks T_4, T_5: have a single ancestor

PastryGrid's Terminology

Task terminology

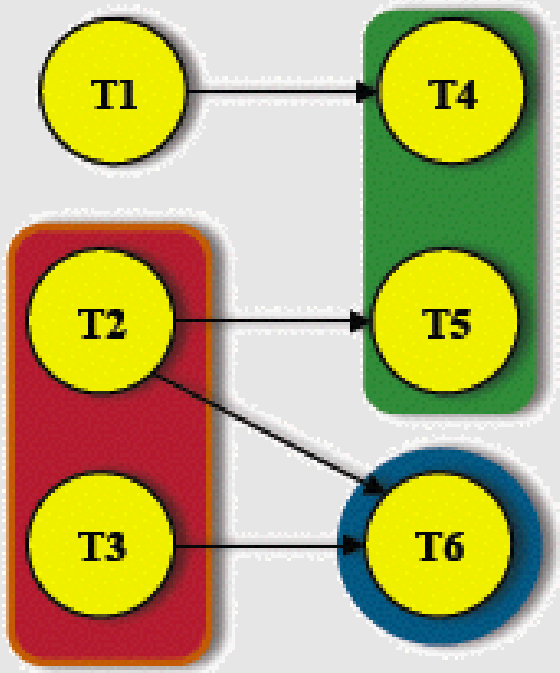
- Friend tasks: T_2, T_3 share the same successor (T_6)
- Shared tasks T_6: has $n > 1$ ancestors (T_2, T_3)
- Isolated tasks T_4, T_5: have a single ancestor

PastryGrid's Terminology

Task terminology

- Friend tasks: T_2, T_3 share the same successor (T_6)
- Shared tasks T_6: has $n > 1$ ancestors (T_2, T_3)
- Isolated tasks T_4, T_5: have a single ancestor
-

Example



PastryGrid components

- Addressing scheme to identify applications and users (based on haching application name + submission date + user name — DHT (Pastry))

PastryGrid components

- Addressing scheme to identify applications and users (based on haching application name + submission date + user name — DHT (Pastry))
- Protocol of resource discovering; No dedicated nodes for the search of the next node to use → on the fly! Optimization: the machine that terminates the last starts the search.

PastryGrid components

- Addressing scheme to identify applications and users (based on haching application name + submission date + user name — DHT (Pastry))
- Protocol of resource discovering; No dedicated nodes for the search of the next node to use → on the fly! Optimization: the machine that terminates the last starts the search.
- Rendez-vous concept (RDV); Objectives: localisation of a node without IP; task coordination; data recovery;

PastryGrid components

- Addressing scheme to identify applications and users (based on haching application name + submission date + user name — DHT (Pastry))
- Protocol of resource discovering; No dedicated nodes for the search of the next node to use → on the fly! Optimization: the machine that terminates the last starts the search.
- Rendez-vous concept (RDV); Objectives: localisation of a node without IP; task coordination; data recovery;
- coordination protocol between machines participating in the application.

PastryGrid components

- Addressing scheme to identify applications and users (based on haching application name + submission date + user name — DHT (Pastry))
- Protocol of resource discovering; No dedicated nodes for the search of the next node to use → on the fly! Optimization: the machine that terminates the last starts the search.
- Rendez-vous concept (RDV); Objectives: localisation of a node without IP; task coordination; data recovery;
- coordination protocol between machines participating in the application.

RDV Concept

Coordinator

- Known at the beginning;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;
- Management of all applications (overload)

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;
- Management of all applications (overload)

RDV

- Unknown;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;
- Management of all applications (overload)

RDV

- Unknown;
- Variable;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;
- Management of all applications (overload)

RDV

- Unknown;
- Variable;
- Failure: may still run;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;
- Management of all applications (overload)

RDV

- Unknown;
- Variable;
- Failure: may still run;
- Distributed data management;

RDV Concept

Coordinator

- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;
- Management of all applications (overload)

RDV

- Unknown;
- Variable;
- Failure: may still run;
- Distributed data management;
- RDV for each application (limited overload)

RDV Concept

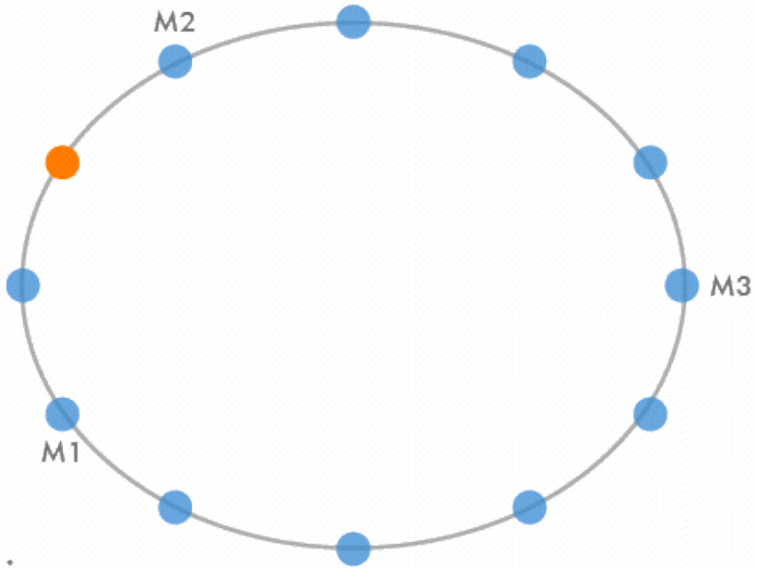
Coordinator

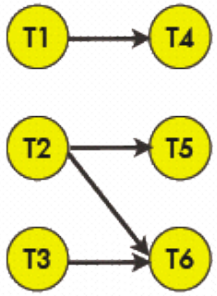
- Known at the beginning;
- Central element on a decicated place;
- Failure: the system crashes;
- Centralized resource management;
- Management of all applications (overload)

RDV

- Unknown;
- Variable;
- Failure: may still run;
- Distributed data management;
- RDV for each application (limited overload)

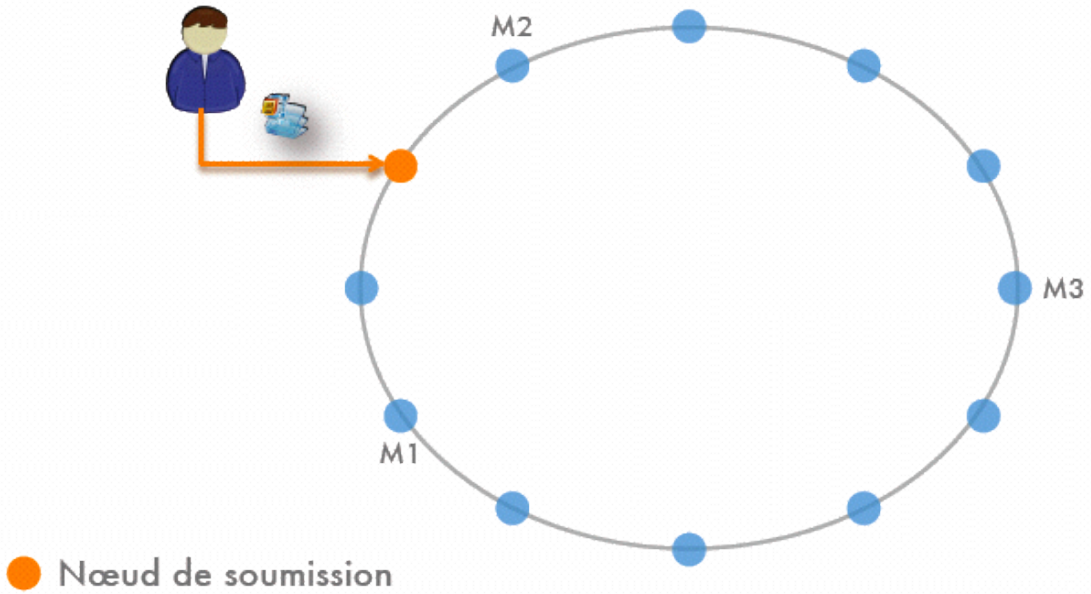
How PastryGrid works

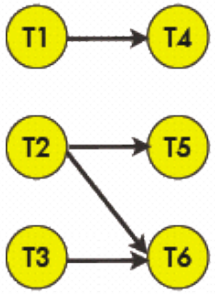




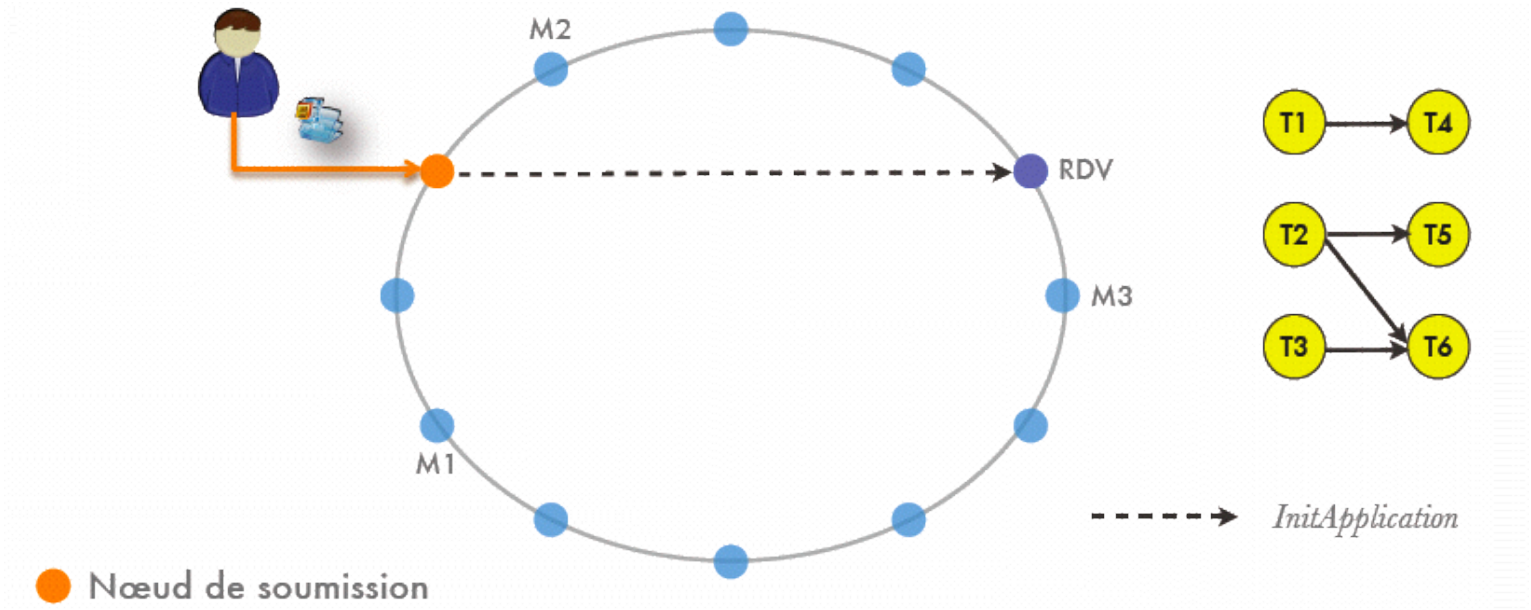
Nœud de soumission

How PastryGrid works

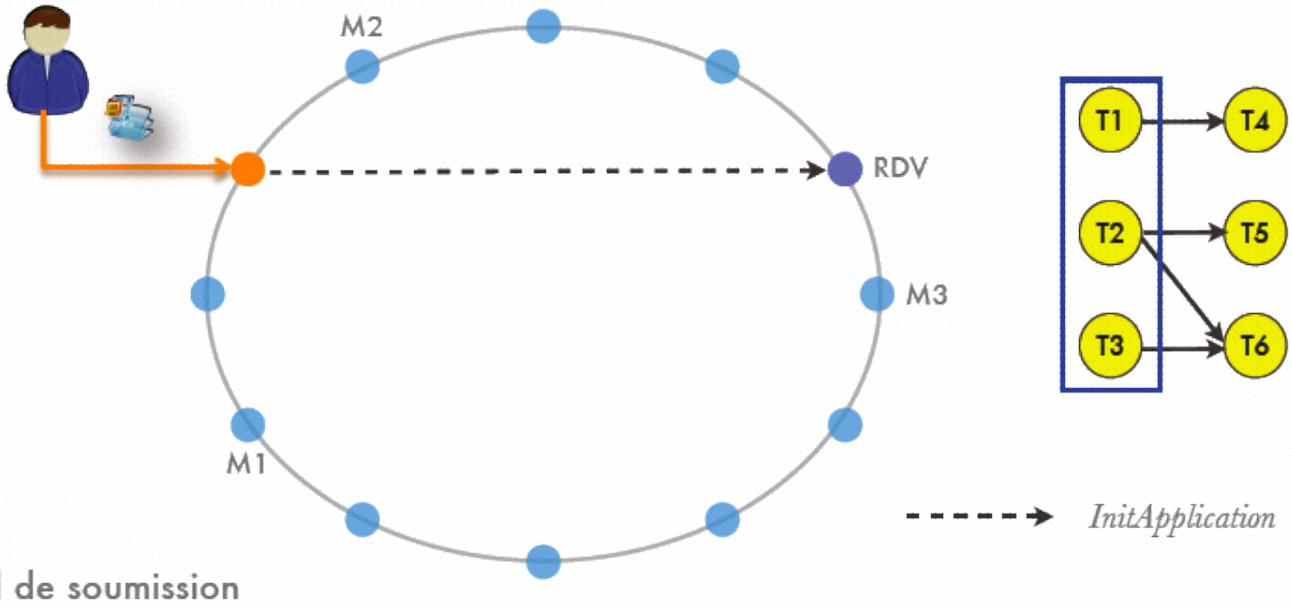




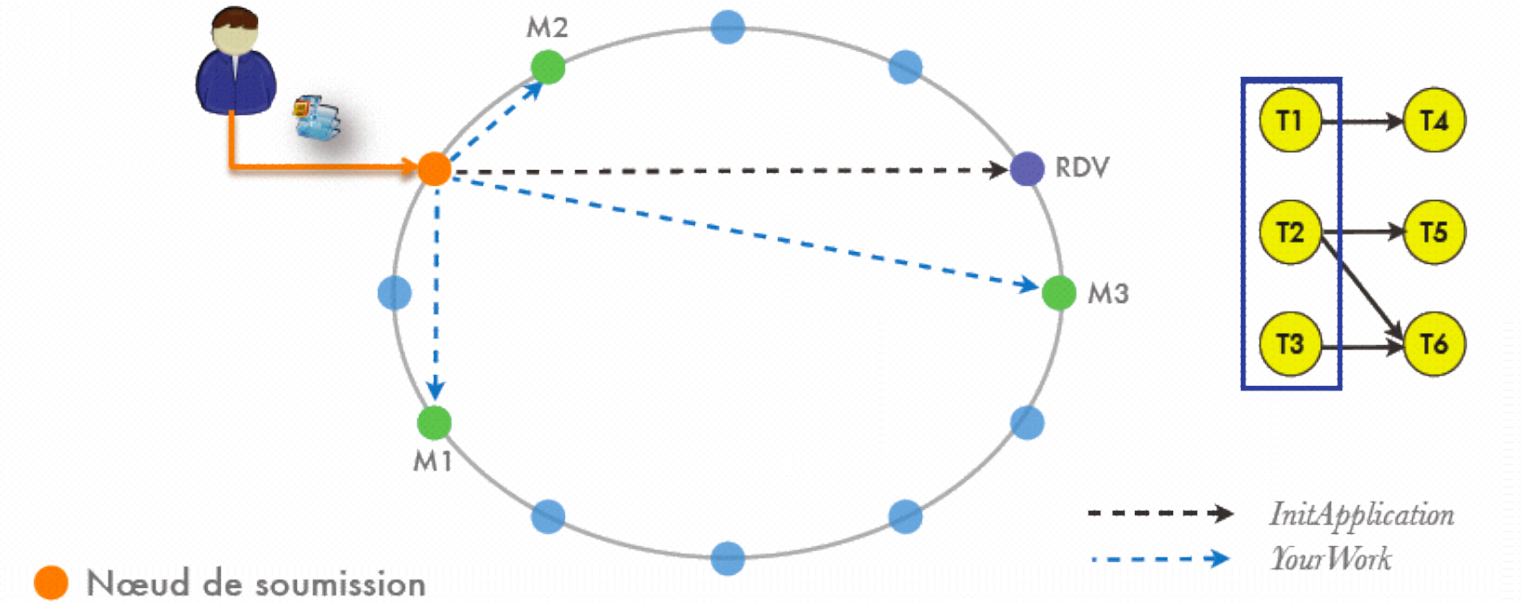
How PastryGrid works



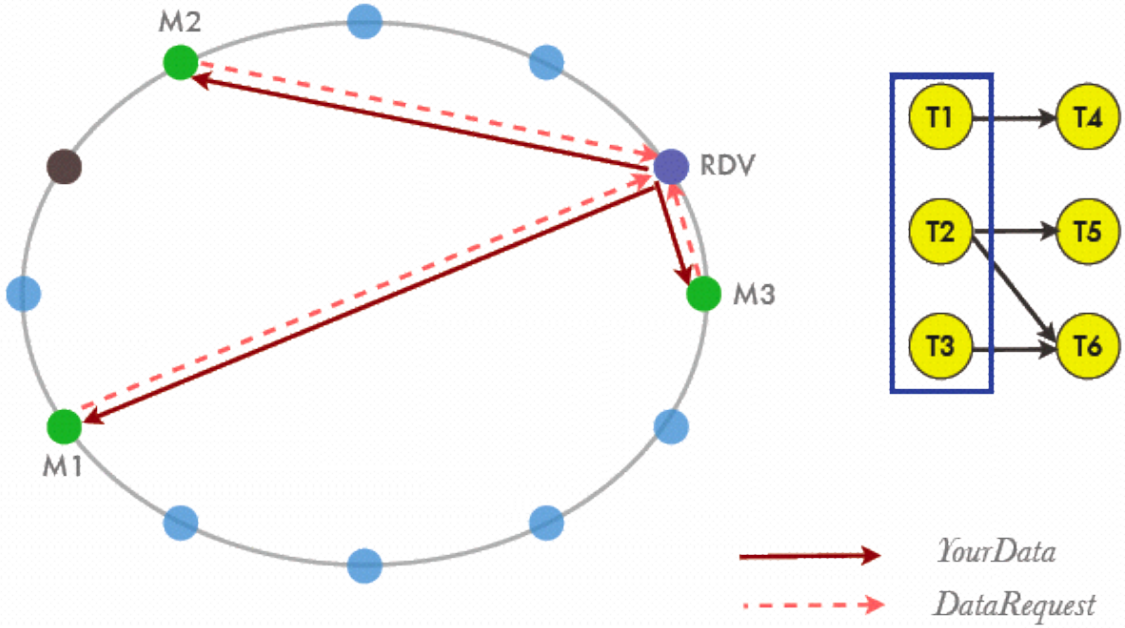
How PastryGrid works



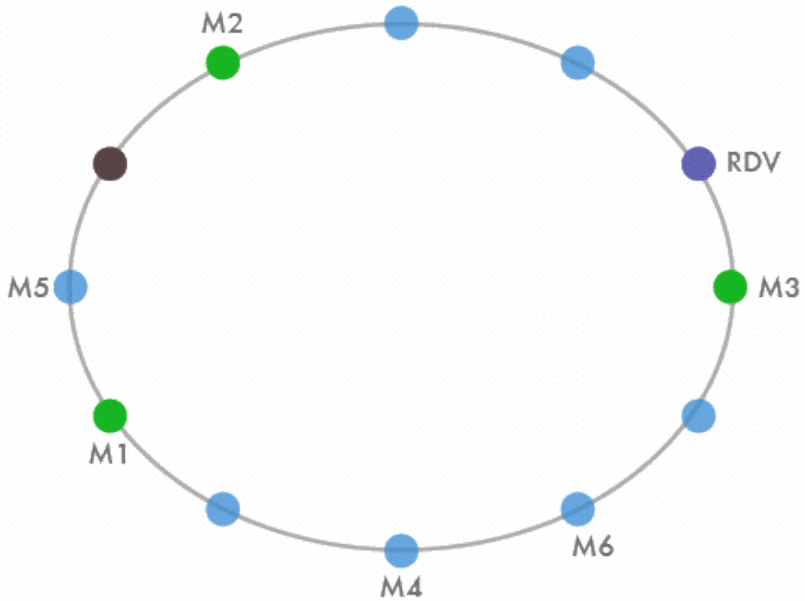
How PastryGrid works

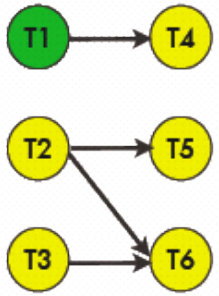


How PastryGrid works

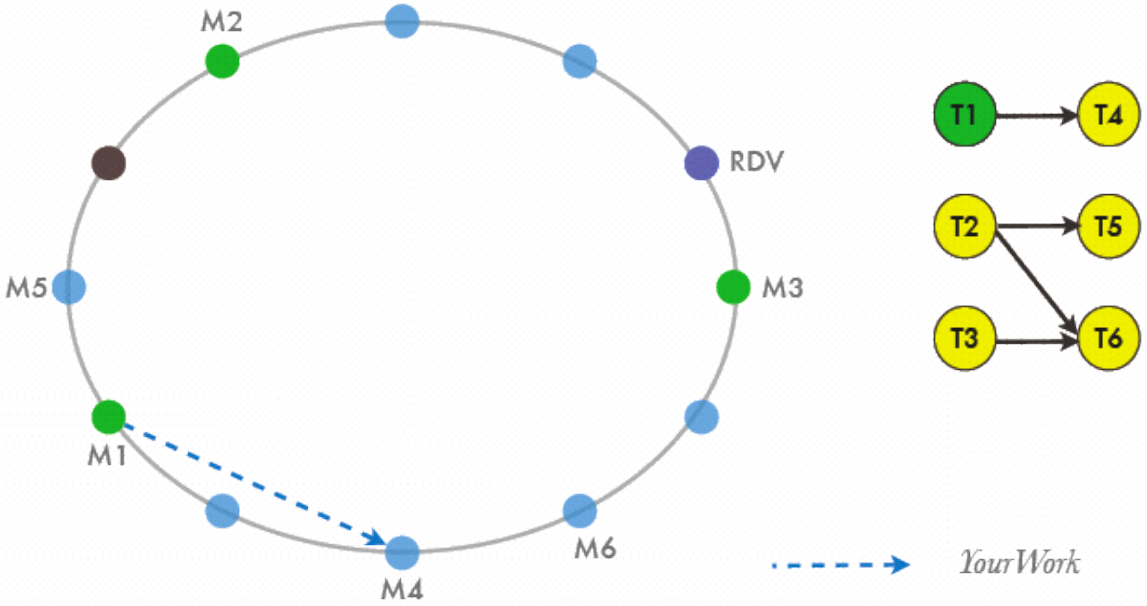


How PastryGrid works

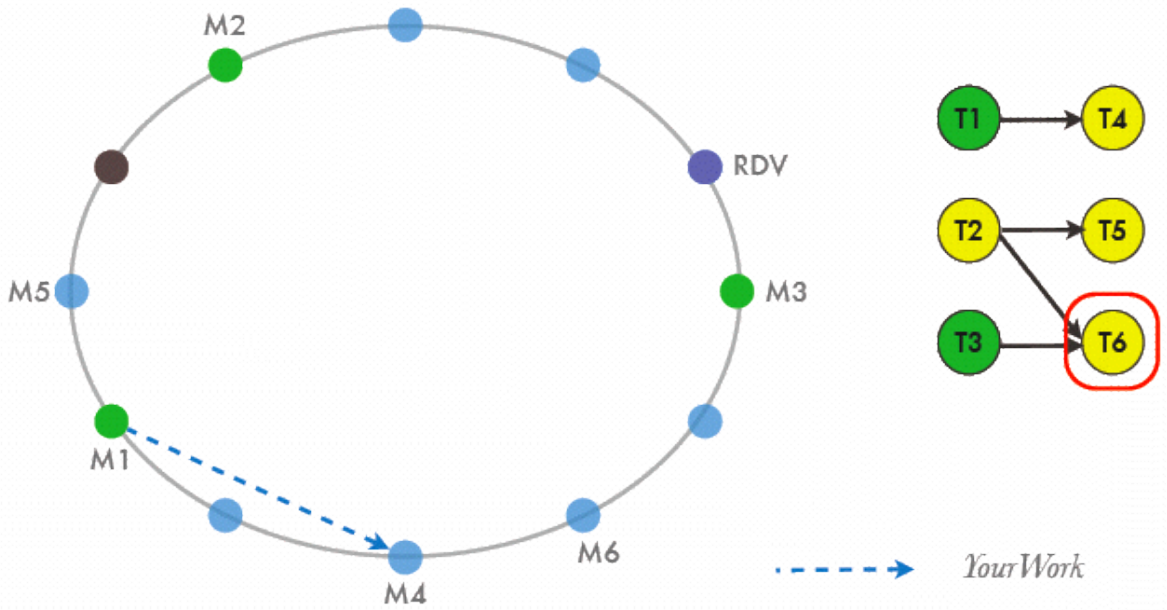




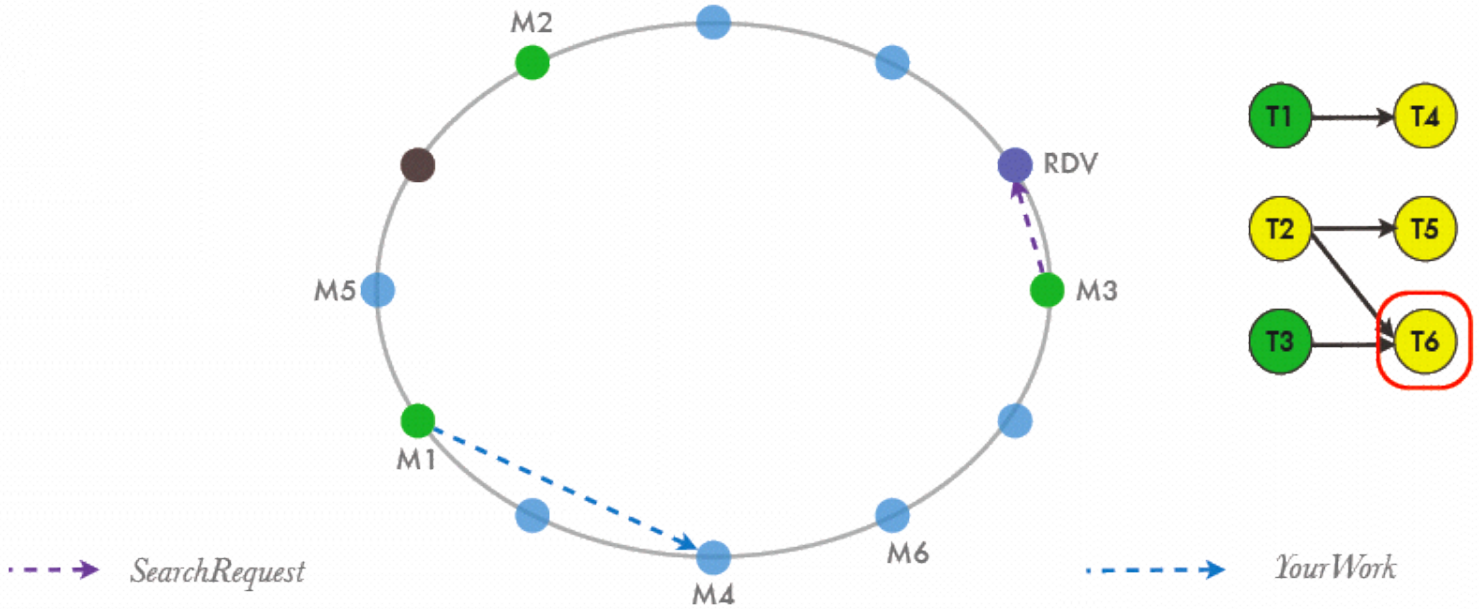
How PastryGrid works



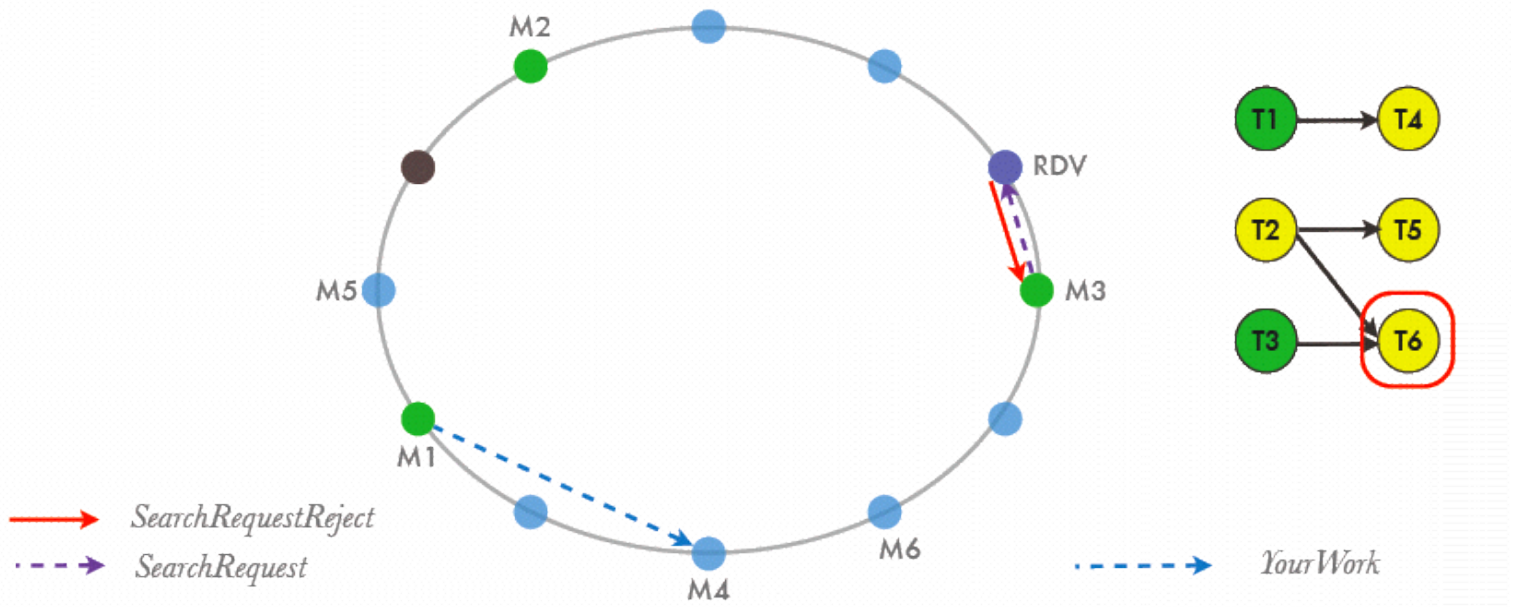
How PastryGrid works



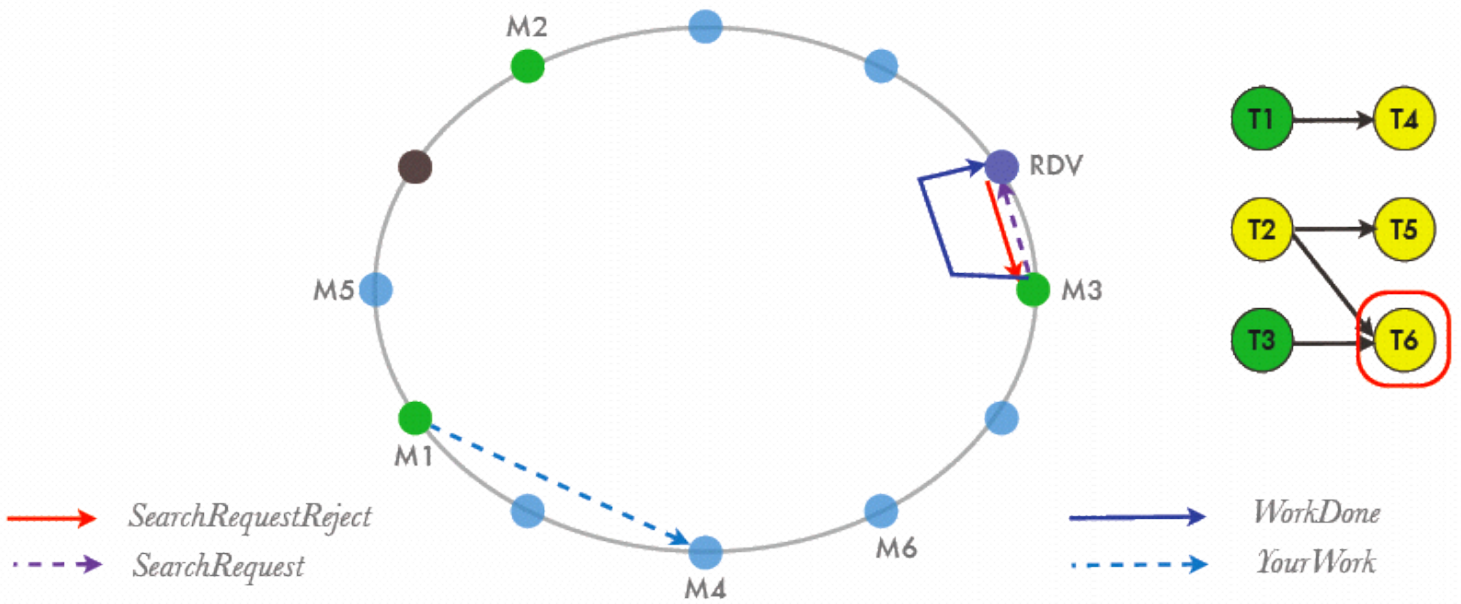
How PastryGrid works



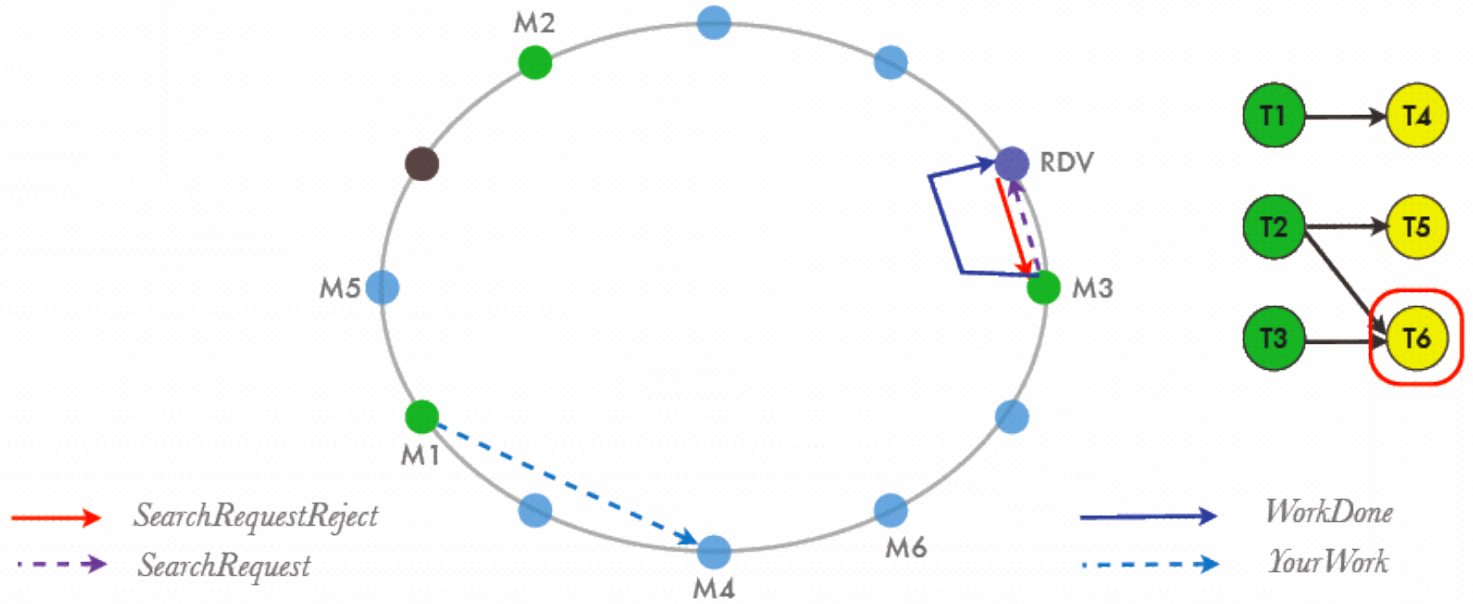
How PastryGrid works



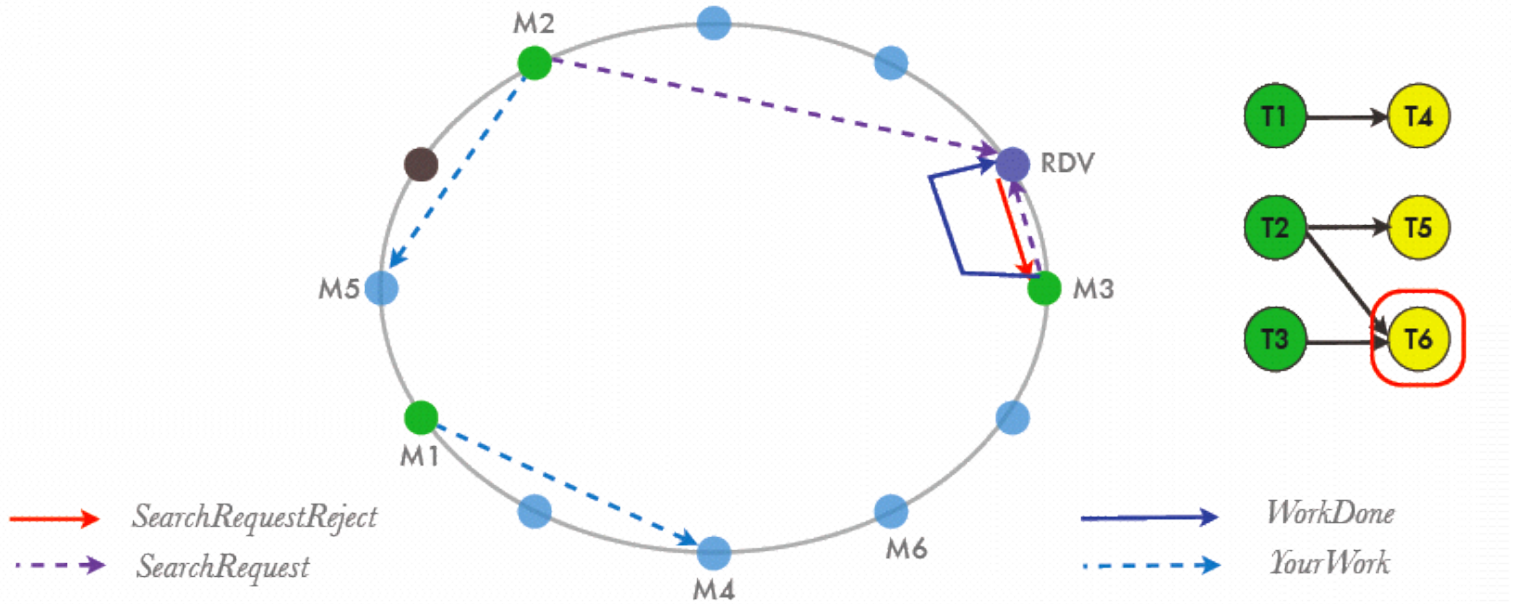
How PastryGrid works



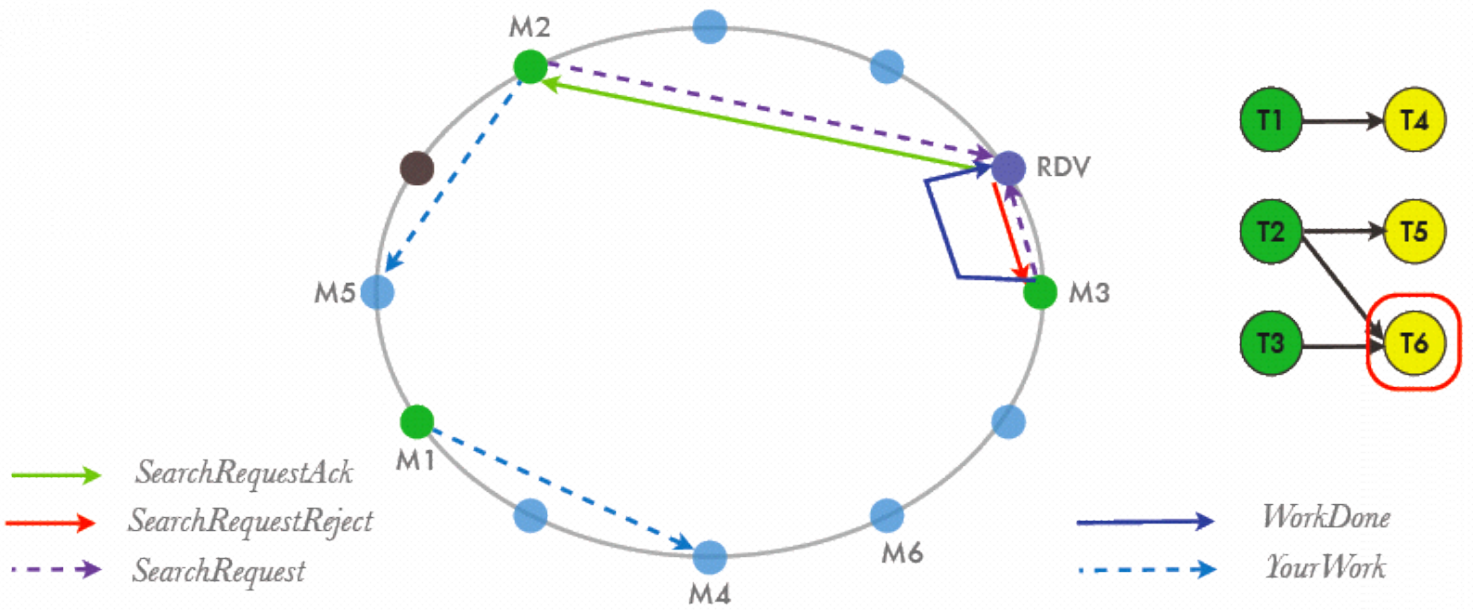
How PastryGrid works



How PastryGrid works



How PastryGrid works



Fault Tolerance in PastryGrid (brief overview)

- Active replication based on Past (maintaining of k copies of the node states) ; update copies when a modification occurs on a source node; automatically creation of a copy (to maintain k)

Fault Tolerance in PastryGrid (brief overview)

- Active replication based on Past (maintaining of k copies of the node states) ; update copies when a modification occurs on a source node; automatically creation of a copy (to maintain k)
- If we adopt such approach $\Rightarrow$ node explosion;

Fault Tolerance in PastryGrid (brief overview)

- Active replication based on Past (maintaining of k copies of the node states) ; update copies when a modification occurs on a source node; automatically creation of a copy (to maintain k)
- If we adopt such approach $\Rightarrow$ node explosion;
- A new component has been added: FTC (Fault Tolerant Component) node
  - Supervises tasks that are running;

Fault Tolerance in PastryGrid (brief overview)

- Active replication based on Past (maintaining of k copies of the node states) ; update copies when a modification occurs on a source node; automatically creation of a copy (to maintain k)
- If we adopt such approach $\Rightarrow$ node explosion;
- A new component has been added: FTC (Fault Tolerant Component) node
  - Supervises tasks that are running;
  - A FCT component for each application; It contacts the RDV to decide the tasks to supervise;

Fault Tolerance in PastryGrid (brief overview)

- Active replication based on Past (maintaining of k copies of the node states) ; update copies when a modification occurs on a source node; automatically creation of a copy (to maintain k)
- If we adopt such approach $\Rightarrow$ node explosion;
- A new component has been added: FTC (Fault Tolerant Component) node
  - Supervises tasks that are running;
  - A FCT component for each application; It contacts the RDV to decide the tasks to supervise;
  - k copies of the FCT and k copies of the RDV per application. In fact you have 3 types of nodes: computing nodes, FCT nodes and RDV nodes to manage;

Fault Tolerance in PastryGrid (brief overview)

- Active replication based on Past (maintaining of k copies of the node states) ; update copies when a modification occurs on a source node; automatically creation of a copy (to maintain k)
- If we adopt such approach $\Rightarrow$ node explosion;
- A new component has been added: FTC (Fault Tolerant Component) node
  - Supervises tasks that are running;
  - A FCT component for each application; It contacts the RDV to decide the tasks to supervise;
  - k copies of the FCT and k copies of the RDV per application. In fact you have 3 types of nodes: computing nodes, FCT nodes and RDV nodes to manage;

PastryGrid Validation

The FT part

- Intensive experiments have been conducted (each machine has a probability P to fail for X seconds): $P = 20\%, 40\%, 80\%$; 100 applications (2 to 128 // tasks) ; on 200 nodes

PastryGrid Validation

The FT part

- Intensive experiments have been conducted (each machine has a probability P to fail for X seconds): $P = 20\%, 40\%, 80\%$; 100 applications (2 to 128 // tasks) ; on 200 nodes
- Main observations:
  - In all cases, PastryGrid terminates;
  - The recovery time depends on the node type;
  - The delay varies from 4:53s to 7:16:41s. . . but it works! The number of delayed applications varies from 44 to 98.

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;
  - No update problem for the user (he always uses the last release);

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

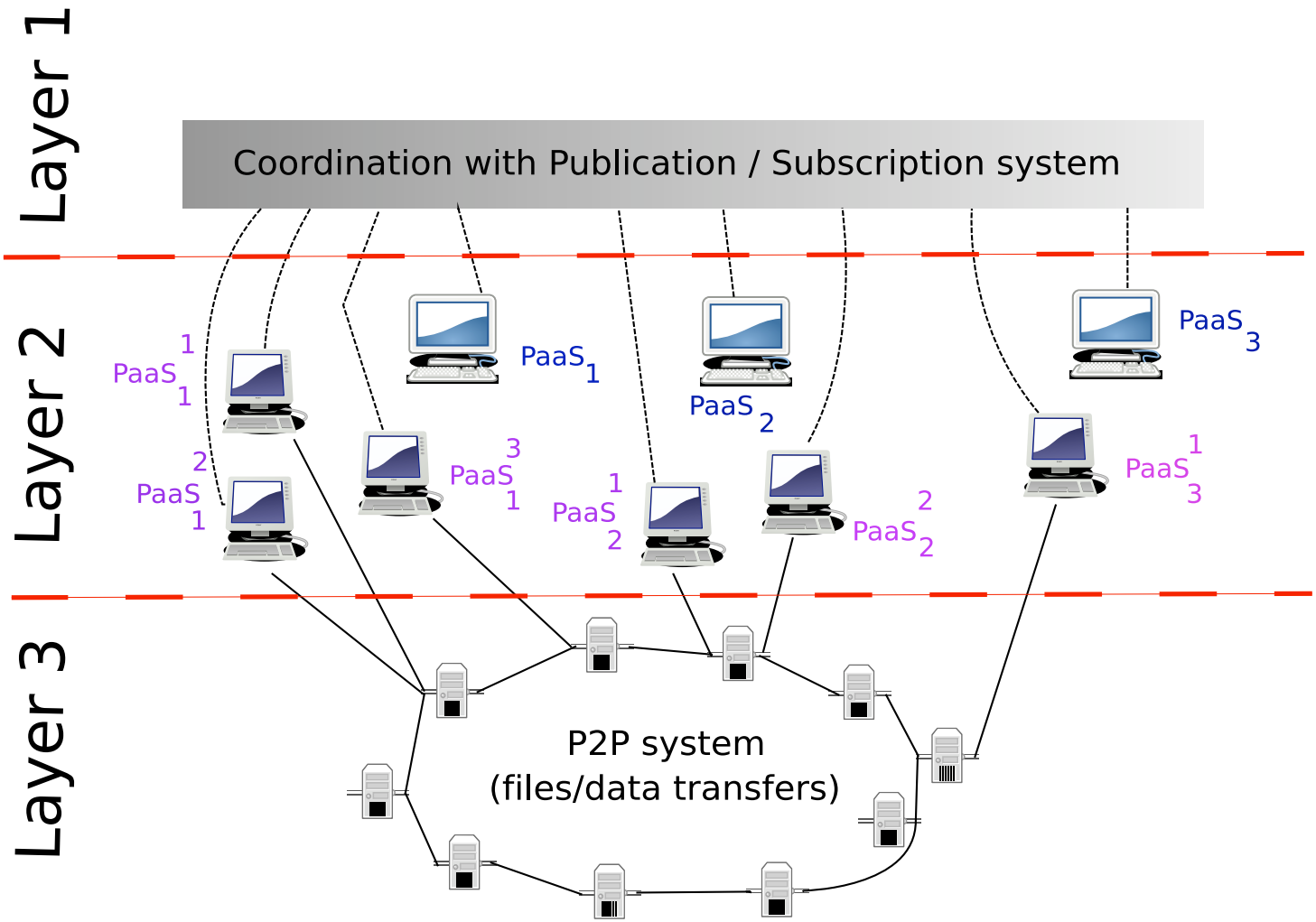
- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;
  - No update problem for the user (he always uses the last release);
  - No maintenance, no local storage.

Towards PaaS and Clouds

The new context: Platform as a Service and Cloud

- Outsourcing of software resources (Google word/spreadsheet online) and hardware resources (Amazon EC2);
- A variant: Platform as a Service (PaaS) where users also inherit from a complete development infrastructure (based on Javascript for the future?);
  - No hosting problem for the user;
  - No update problem for the user (he always uses the last release);
  - No maintenance, no local storage.
- We have started an initiative for defining and designing a "general purpose PaaS" based on distributed protocols for coordination and data exchange.

Architecture overview of the PaaSoordinated project



Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally ⇒ an open protocol does not capture the user

Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally ⇒ an open protocol does not capture the user
- Different instances (say, of an ERP) do public announcements to search for providers, then explore HTML links (interrogate different Clouds able to answer) ⇒ an open protocol do not capture the user, again

Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally ⇒ an open protocol does not capture the user
- Different instances (say, of an ERP) do public announcements to search for providers, then explore HTML links (interrogate different Clouds able to answer) ⇒ an open protocol do not capture the user, again
- So, we want open protocols to coordinate and to exchange data!

Key points regarding Philosophy

- Different instances (say, of a database) want to exchange data temporally ⇒ an open protocol does not capture the user
- Different instances (say, of an ERP) do public announcements to search for providers, then explore HTML links (interrogate different Clouds able to answer) ⇒ an open protocol do not capture the user, again
- So, we want open protocols to coordinate and to exchange data!

Some Challenges

- Where to insert the different connectors in the PaaS software stack to get an open infrastructure?

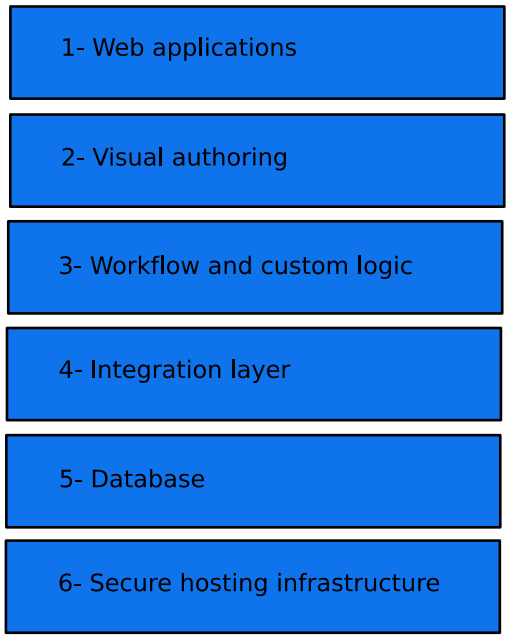


Figure: Software stack in PaaS (source: Coghead)

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service → mastering FT → redundancy;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service $\rightarrow$ mastering FT $\rightarrow$ redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service $\rightarrow$ mastering FT $\rightarrow$ redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks $\rightarrow$ use P2P techniques;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service → mastering FT → redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks → use P2P techniques;
- Bugs in Large-Scale Distributed Systems → use Formal Methods to specify and to analyse architecture and protocols;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service → mastering FT → redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks → use P2P techniques;
- Bugs in Large-Scale Distributed Systems → use Formal Methods to specify and to analyse architecture and protocols;
- Scaling Quickly → instanciate new PaaS on the fly → define a model for cooperation and interaction;

Research opportunities

Above the Clouds: A Berkeley View of Cloud Computing

- Availability of a service $\rightarrow$ mastering FT $\rightarrow$ redundancy;
- Data Lock-In: API must not be proprietary but should rely on open standards;
- Data Transfer Bottlenecks $\rightarrow$ use P2P techniques;
- Bugs in Large-Scale Distributed Systems $\rightarrow$ use Formal Methods to specify and to analyse architecture and protocols;
- Scaling Quickly $\rightarrow$ instanciate new PaaS on the fly $\rightarrow$ define a model for cooperation and interaction;

State of the Art

Similar projects

- Vertebra (http://www.engineyard.com/): "Service-Oriented-Architecture for the cloud, an application deployment platform" – based on Ruby and Erlang. The project moves to "On-demand deployment and management of your Ruby on Rails applications with Engine Yard Cloud – One-click code deploys, application cloning, data automation. . ."

State of the Art

Similar projects

- Vertebra (http://www.engineyard.com/): "Service-Oriented-Architecture for the cloud, an application deployment platform" – based on Ruby and Erlang. The project moves to "On-demand deployment and management of your Ruby on Rails applications with Engine Yard Cloud – One-click code deploys, application cloning, data automation. . ."
- Kandula (http://ws.apache.org/kandula/): Presently Kandula implements WS-Coordination and WS-AtomicTransaction protocols. WS-BusinessActivity protocol will be available in the near future.
  Kandula project has 2 branches. Kandula1 branch runs on Apache Axis 1.x. Kandula2 branch (new) runs on Apache Axis2

State of the Art

Similar projects

- Vertebra (http://www.engineyard.com/): "Service-Oriented-Architecture for the cloud, an application deployment platform" – based on Ruby and Erlang. The project moves to "On-demand deployment and management of your Ruby on Rails applications with Engine Yard Cloud – One-click code deploys, application cloning, data automation. . ."
- Kandula (http://ws.apache.org/kandula/): Presently Kandula implements WS-Coordination and WS-AtomicTransaction protocols. WS-BusinessActivity protocol will be available in the near future.
  Kandula project has 2 branches. Kandula1 branch runs on Apache Axis 1.x. Kandula2 branch (new) runs on Apache Axis2

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C
- Corona: A High Performance Publish-Subscribe System for the World (http://www.truststc.org/pubs/38.html). Topic based publish subscribe system for the Web. URLs of Web content serve as topics or channels. Any Web object identifiable by an URL can be monitored with Corona

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C
- Corona: A High Performance Publish-Subscribe System for the World (http://www.trustst c.org/pubs/38.html). Topic based publish subscribe system for the Web. URLs of Web content serve as topics or channels. Any Web object identifiable by an URL can be monitored with Corona
- UDDI, XMMP...

State of the Art

Similar projects

- Axis (http://ws.apache.org/axis2/): Axis2 is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. There are two implementations of the Apache Axis2 Web services engine - Apache Axis2/Java and Apache Axis2/C
- Corona: A High Performance Publish-Subscribe System for the World (http://www.trustst c.org/pubs/38.html). Topic based publish subscribe system for the Web. URLs of Web content serve as topics or channels. Any Web object identifiable by an URL can be monitored with Corona
- UDDI, XMMP...

State of the Art

Pieces of the maze (not exhaustive)

- Data exchange: Bitdew (no comment). Comment: have a look to SyncML Protocol (http://www.openmobilealliance.org/syncml/): This open standard seeks to drive data mobility by establishing a common language for communications among devices, applications, and networks.

State of the Art

Pieces of the maze (not exhaustive)

- Data exchange: Bitdew (no comment). Comment: have a look to SyncML Protocol (http://www.openmobilealliance.org/syncml/): This open standard seeks to drive data mobility by establishing a common language for communications among devices, applications, and networks.
- TioLive (http://www.tiolive.com): Open source by Nexedi Corp. for Communication: email, telephone, chat, Backoffice: contacts, documents, accounting, ERP, CRM, e-Business: web site, e-Commerce:

State of the Art

Pieces of the maze (not exhaustive)

- Data exchange: Bitdew (no comment). Comment: have a look to SyncML Protocol (http://www.openmobilealliance.org/syncml/): This open standard seeks to drive data mobility by establishing a common language for communications among devices, applications, and networks.
- TioLive (http://www.tiolive.com): Open source by Nexedi Corp. for Communication: email, telephone, chat, Backoffice: contacts, documents, accounting, ERP, CRM, e-Business: web site, e-Commerce:

State of the Art

Pieces of the maze (not exhaustive)

- Programming: https://www.myerp5.com/kb/developer-How.To.Become.ERP5.Developer/view
- OSOE course: http://www.osoe-project.org/lesson
- TioLive tutorial: https://www.tiolive.com/documentation/tiolive-tutorial
- Documentation for developers:

  https://www.myerp5.com/kb/documentation_section/developer/

  https://www.myerp5.com/kb/documentation_section/developer/developer-Technology/view

  https://www.myerp5.com/kb/documentation_section/developer/enterprise-High..Performance.Zope/view

Conclusion

Hope

- DG has proved to be relevant for resource sharing ⇒ transpose this success story to the Cloud and PaaS universes ⇒ offer a technical alternate to Google, Salesforce, Amazon big farm of servers

Conclusion

Hope

- DG has proved to be relevant for resource sharing ⇒ transpose this success story to the Cloud and PaaS universes ⇒ offer a technical alternate to Google, Salesforce, Amazon big farm of servers
- Our approaches are based on emerging open source Cloud solution. From an economic point of view: if it is less expensive to host services locally and if it offers more advantages (we are not "dependant on a technology" → no prison, more potential partners), then small/medium size companies will adopt our approaches;

Conclusion

Hope

- DG has proved to be relevant for resource sharing ⇒ transpose this success story to the Cloud and PaaS universes ⇒ offer a technical alternate to Google, Salesforce, Amazon big farm of servers
- Our approaches are based on emerging open source Cloud solution. From an economic point of view: if it is less expensive to host services locally and if it offers more advantages (we are not "dependant on a technology" → no prison, more potential partners), then small/medium size companies will adopt our approaches;
- Main change: accept to manage redundancy, scaling the server (even for temporary needs), synchronisation ⇒ coordination with grid technology (BonjourGrid, PastryGrid?);

Conclusion

Hope

- Benefit: less expensive (comparing to Amazone EC2) because you control your data

Conclusion

Hope

- Benefit: less expensive (comparing to Amazone EC2) because you control your data
- Could also be implemented and deployed with a centralized Web Services $\Leftrightarrow$ for each application and for each Cloud type, you need a specific coordination protocol $\Rightarrow$ single point of failure.

Conclusion

Hope

- Benefit: less expensive (comparing to Amazone EC2) because you control your data
- Could also be implemented and deployed with a centralized Web Services $\Leftrightarrow$ for each application and for each Cloud type, you need a specific coordination protocol $\Rightarrow$ single point of failure.
- Ex: a company wants to install the Virtual Desktop EyeOS and the TioLive/ERP5 PaaS. During the night, the company rents different services:
  - one (company) to many – many (services) to many (companies) = new abilities, new business!
  - demonstrate that a single coordination protocol is better than configuring as many middlewares than we have software!

Towards PaaS and Clouds Our experience with BonjourGrid and PastryGrid – AOC Team –

Christophe Cérin[1]

[1]Université de Paris XIII, CNRS UMR 7030, France

Bi-lateral China-France workshop