

# JavaGAT Tutorial

## *Getting started with the Grid Application Toolkit*

Rob van Nieuwpoort, Roelof Kemp

rob@cs.vu.nl, rkemp@cs.vu.nl

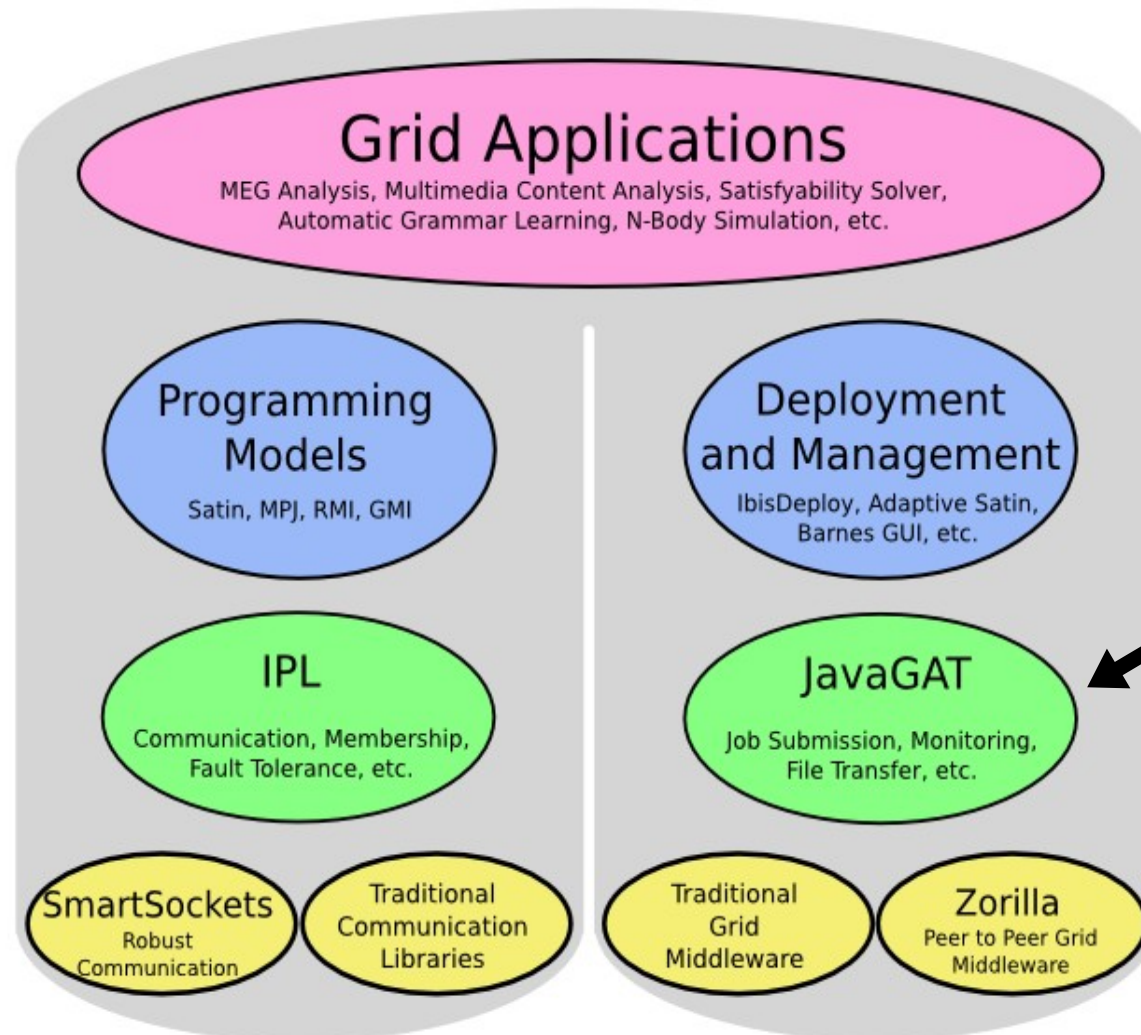
Vrije Universiteit  
Amsterdam



vrije Universiteit

Rob van Nieuwpoort

- What is GAT and why do we need it?
- JavaGAT structure and overview
- Security
- Grid I/O



## **Grid Application**

`File.copy(...)`



**Grid Application**

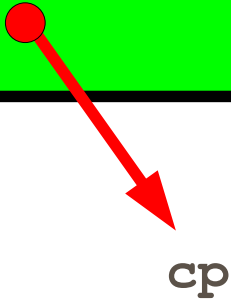
`submitJob(...)`



`File.copy(...)`

**Grid Application**

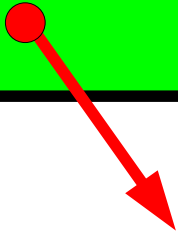
`submitJob(...)`



`File.copy(...)`

## **Grid Application**

`submitJob(...)`

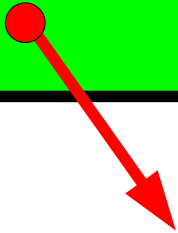


`cp`  
`ftp`

`File.copy(...)`

## **Grid Application**

`submitJob(...)`



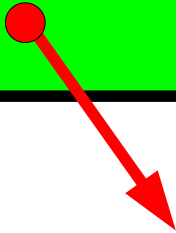
`cp`  
`ftp`  
`gridftp`



`File.copy(...)`

## **Grid Application**

`submitJob(...)`



`cp`  
`ftp`  
`gridftp`  
`scp`

`File.copy(...)`

## Grid Application

`submitJob(...)`



cp  
ftp  
gridftp  
scp  
http



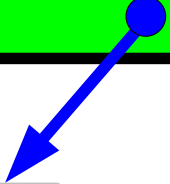
`File.copy(...)`

## Grid Application

`submitJob(...)`

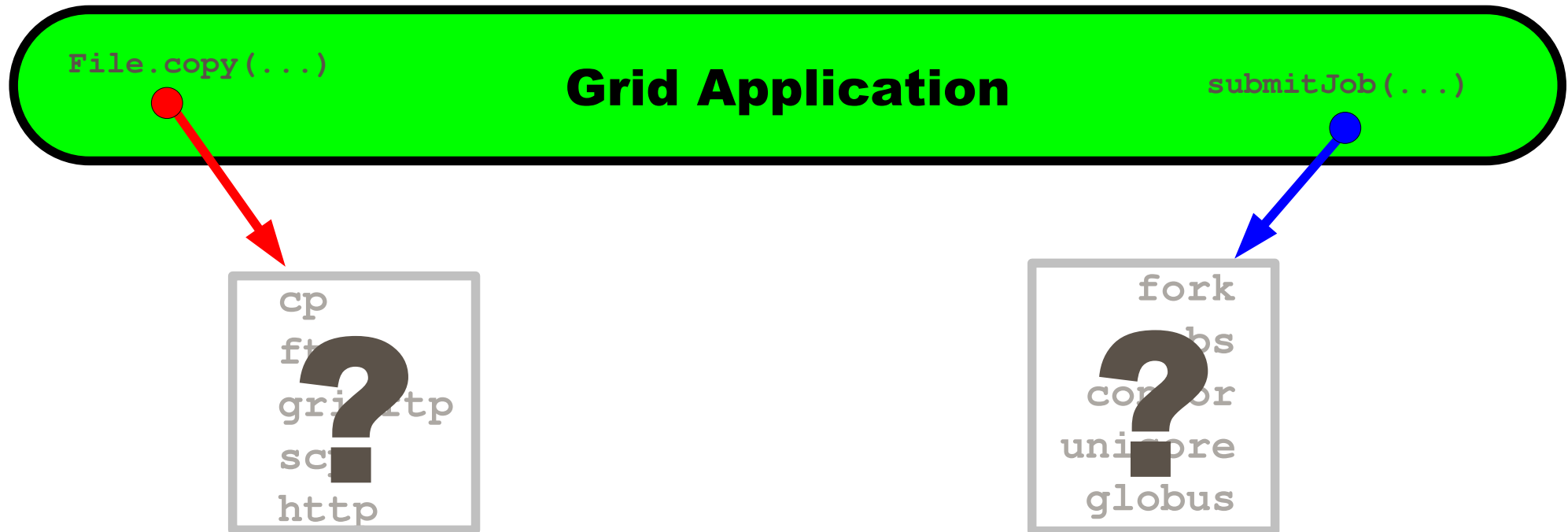


cp  
ftp  
gridftp  
scp  
http

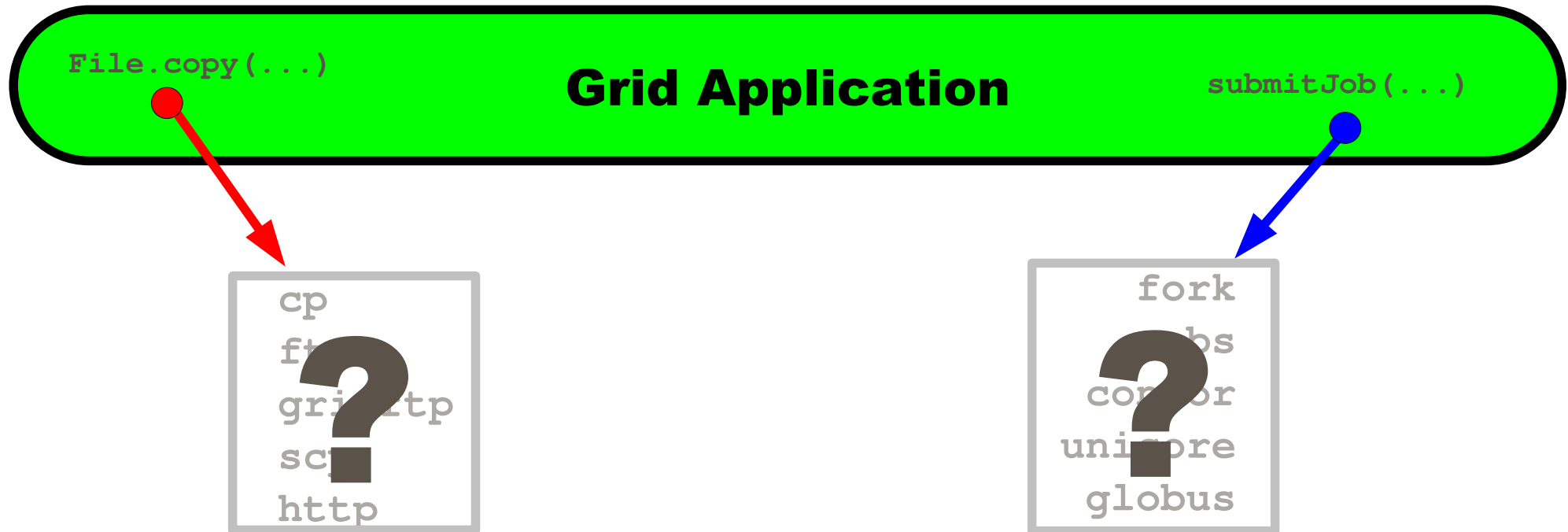


fork  
pbs  
com or  
unipore  
globus

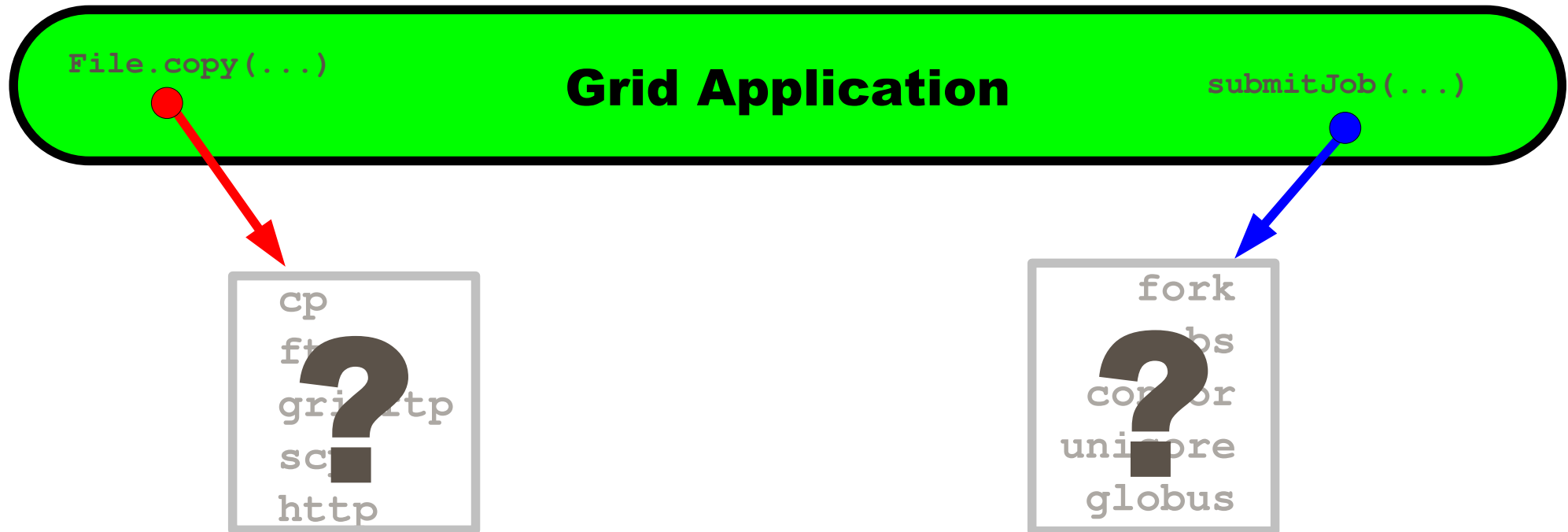
- Which should you use?



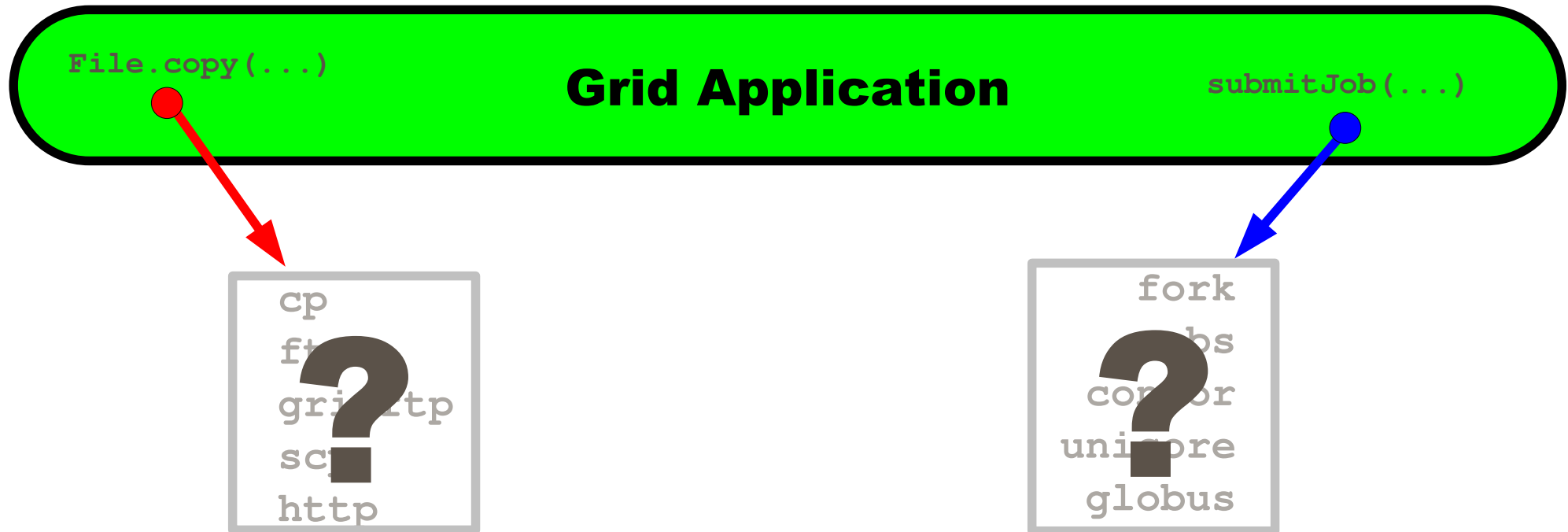
- Which should you use?
- Some might not be available on all sites



- Which should you use?
- Some might not be available on all sites
- Some may not work for all users (certificates)

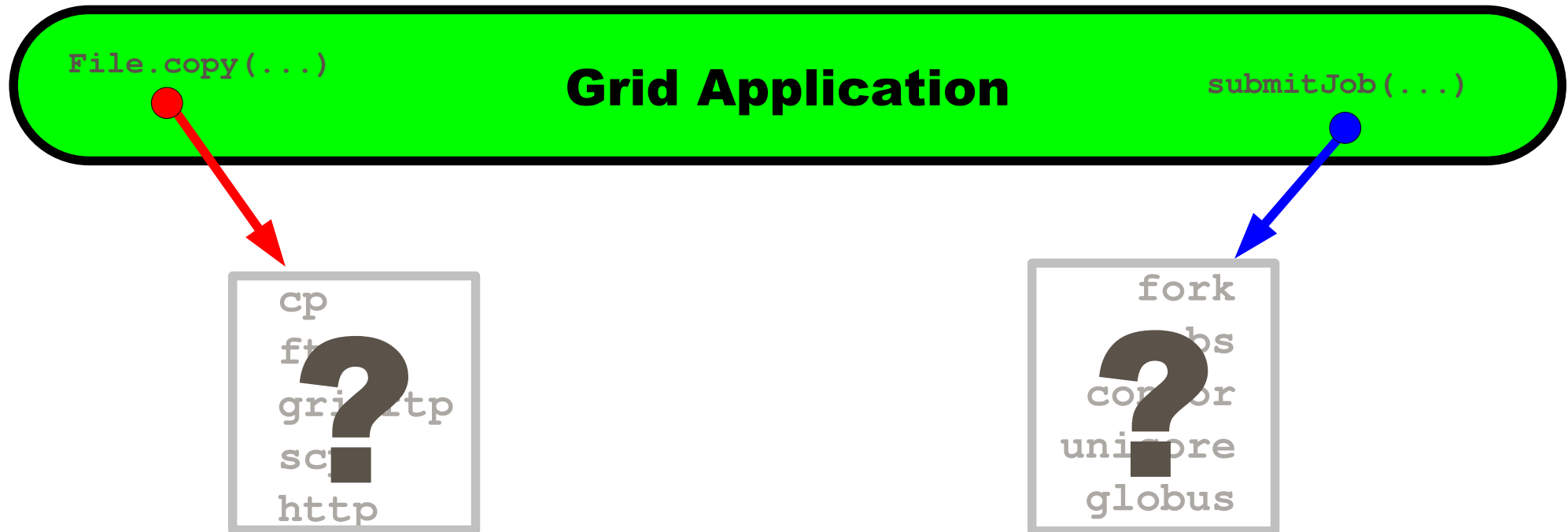


- Which should you use?
- Some might not be available on all sites
- Some may not work for all users (certificates)
- Version differences (Globus changes every 2 months)



- Which should you use?
- Some might not be available on all sites
- Some may not work for all users (certificates)
- Version differences (Globus changes every 2 months)
- Exponential number of combinations!





- Which should you use?
- Some might not be available on all sites
- Some may not work for all users (security context)
- Version differences (Globus changes every 2 months)
- Exponential number of combinations!

# Not portable!

```
int RemoteFile::GetFile (char const* source,  char const* target) {
    globus_url_t          source_url;
    globus_io_handle_t     dest_io_handle;
    globus_ftp_client_operationattr_t  source_ftp_attr;
    globus_result_t        result;
    globus_gass_transfer_requestattr_t source_gass_attr;
    globus_gass_copy_attr_t source_gass_copy_attr;
    globus_gass_copy_handle_t gass_copy_handle;
    globus_gass_copy_handleattr_t gass_copy_handleattr;
    globus_ftp_client_handleattr_t ftp_handleattr;
    globus_io_attr_t        io_attr;
    int                     output_file = -1;

    if ( globus_url_parse (source_URL, &source_url) != GLOBUS_SUCCESS ) {
        printf ("can not parse source_URL \"%s\"\n", source_URL);
        return (-1);
    }

    if ( source_url.scheme_type != GLOBUS_URL_SCHEME_GSIFTP &&
        source_url.scheme_type != GLOBUS_URL_SCHEME_FTP    &&
        source_url.scheme_type != GLOBUS_URL_SCHEME_HTTP   &&
        source_url.scheme_type != GLOBUS_URL_SCHEME_HTTPS  ) {
        printf ("can not copy from %s - wrong prot\n", source_URL);
        return (-1);
    }

    globus_gass_copy_handleattr_init (&gass_copy_handleattr);
    globus_gass_copy_attr_init      (&source_gass_copy_attr);

    globus_ftp_client_handleattr_init (&ftp_handleattr);
    globus_io_fileattr_init           (&io_attr);

    globus_gass_copy_attr_set_io      (&source_gass_copy_attr, &io_attr);
    globus_gass_copy_handleattr_set_ftp_attr
        (&gass_copy_handleattr,
         &ftp_handleattr);

    globus_gass_copy_handle_init      (&gass_copy_handle,
        &gass_copy_handleattr);
}
```

```
if (source_url.scheme_type == GLOBUS_URL_SCHEME_GSIFTP ||
    source_url.scheme_type == GLOBUS_URL_SCHEME_FTP    ) {
    globus_ftp_client_operationattr_init (&source_ftp_attr);
    globus_gass_copy_attr_set_ftp (&source_gass_copy_attr,
        &source_ftp_attr);
}
else {
    globus_gass_transfer_requestattr_init (&source_gass_attr,
        source_url.scheme);
    globus_gass_copy_attr_set_gass (&source_gass_copy_attr,
        &source_gass_attr);
}

output_file = globus_libc_open ((char*) target,
    O_WRONLY | O_TRUNC | O_CREAT,
    S_IRUSR | S_IWUSR | S_IRGRP |
    S_IWGRP);

if ( output_file == -1 ) {
    printf ("could not open the file \"%s\"\n", target);
    return (-1);
}

/* convert stdout to be a globus_io_handle */
if ( globus_io_file_posix_convert (output_file, 0,
        &dest_io_handle)
    != GLOBUS_SUCCESS) {
    printf ("Error converting the file handle\n");
    return (-1);
}

result = globus_gass_copy_register_url_to_handle (
    &gass_copy_handle, (char*)source_URL,
    &source_gass_copy_attr, &dest_io_handle,
    my_callback, NULL);

if ( result != GLOBUS_SUCCESS ) {
    printf ("error: %s\n", globus_object_printable_to_string
        (globus_error_get (result)));
    return (-1);
}

globus_url_destroy (&source_url);
return (0);
}
```



# CoG/RFT File copy (C++)



```
package org.globus.ogsa.gui;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.net.URL;
import java.util.Date;
import java.util.Vector;
import javax.xml.rpc.Stub;
import org.apache.axis.message.MessageElement;
import org.apache.axis.utils.XMLUtils;
import org.globus.*
import org.gridforum.ogsi.*
import org.gridforum.ogsi.holders.TerminationTimeTypeHolder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class RFTClient {
    public static void copy (String source_url, String target_url) {
        try {
            File requestFile = new File (source_url);
            BufferedReader reader = null;
            try {
                reader = new BufferedReader (new FileReader (requestFile));
            } catch (java.io.FileNotFoundException fnfe) { }
            Vector requestData = new Vector ();
            requestData.add (target_url);
            TransferType[] transfers1 = new TransferType[transferCount];
            RFTOptionsType multirftOptions = new RFTOptionsType ();

            multirftOptions.setBinary (Boolean.valueOf (
                (String)requestData.elementAt (0)).booleanValue ());
            multirftOptions.setBlockSize (Integer.valueOf (
                (String)requestData.elementAt (1)).intValue ());
            multirftOptions.setTcpBufferSize (Integer.valueOf (
                (String)requestData.elementAt (2)).intValue ());
            multirftOptions.setNotpt (Boolean.valueOf (
                (String)requestData.elementAt (3)).booleanValue ());
            multirftOptions.setParallelStreams (Integer.valueOf (
                (String)requestData.elementAt (4)).intValue ());
            multirftOptions.setDcau(Boolean.valueOf(
                (String)requestData.elementAt (5)).booleanValue ());

            int i = 7;
            for (int j = 0; j < transfers1.length; j++)
            {
                transfers1[j] = new TransferType ();

                transfers1[j].setTransferId (j);
                transfers1[j].setSourceUrl ((String)requestData.elementAt (i++));
                transfers1[j].setDestinationUrl ((String)requestData.elementAt (i++));
                transfers1[j].setRftOptions (multirftOptions);
            }
        }
    }
}
```

```
TransferRequestType transferRequest = new TransferRequestType ();
transferRequest.setTransferArray (transfers1);

int concurrency = Integer.valueOf
    ((String)requestData.elementAt(6)).intValue();

if (concurrency > transfers1.length)
{
    System.out.println ("Concurrency should be less than the number"
        "of transfers in the request");
    System.exit (0);
}
transferRequest.setConcurrency (concurrency);

TransferRequestElement requestElement = new TransferRequestElement ();
requestElement.setTransferRequest (transferRequest);

ExtensibilityType extension = new ExtensibilityType ();
extension = AnyHelper.getExtensibility (requestElement);

OGSIServiceGridLocator factoryService = new OGSIServiceGridLocator ();
Factory factory = factoryService.getFactoryPort (new URL (source_url));
GridServiceFactory gridFactory = new GridServiceFactory (factory);

LocatorType locator = gridFactory.createService (extension);
System.out.println ("Created an instance of Multi-RFT");

MultiFileRFTDefinitionServiceGridLocator loc
    = new MultiFileRFTDefinitionServiceGridLocator ();
RFTPortType rftPort = loc.getMultiFileRFTDefinitionPort (locator);
((Stub)rftPort)._setProperty (Constants.AUTHORIZATION,
    NoAuthorization.getInstance());
((Stub)rftPort)._setProperty (GSIConstants.GSI_MODE,
    GSIConstants.GSI_MODE_FULL_DELEG);
((Stub)rftPort)._setProperty (Constants.GSI_SEC_CONV,
    Constants.SIGNATURE);
((Stub)rftPort)._setProperty (Constants.GRIM_POLICY_HANDLER,
    new IgnoreProxyPolicyHandler ());

int requestid = rftPort.start ();
System.out.println ("Request id: " + requestid);

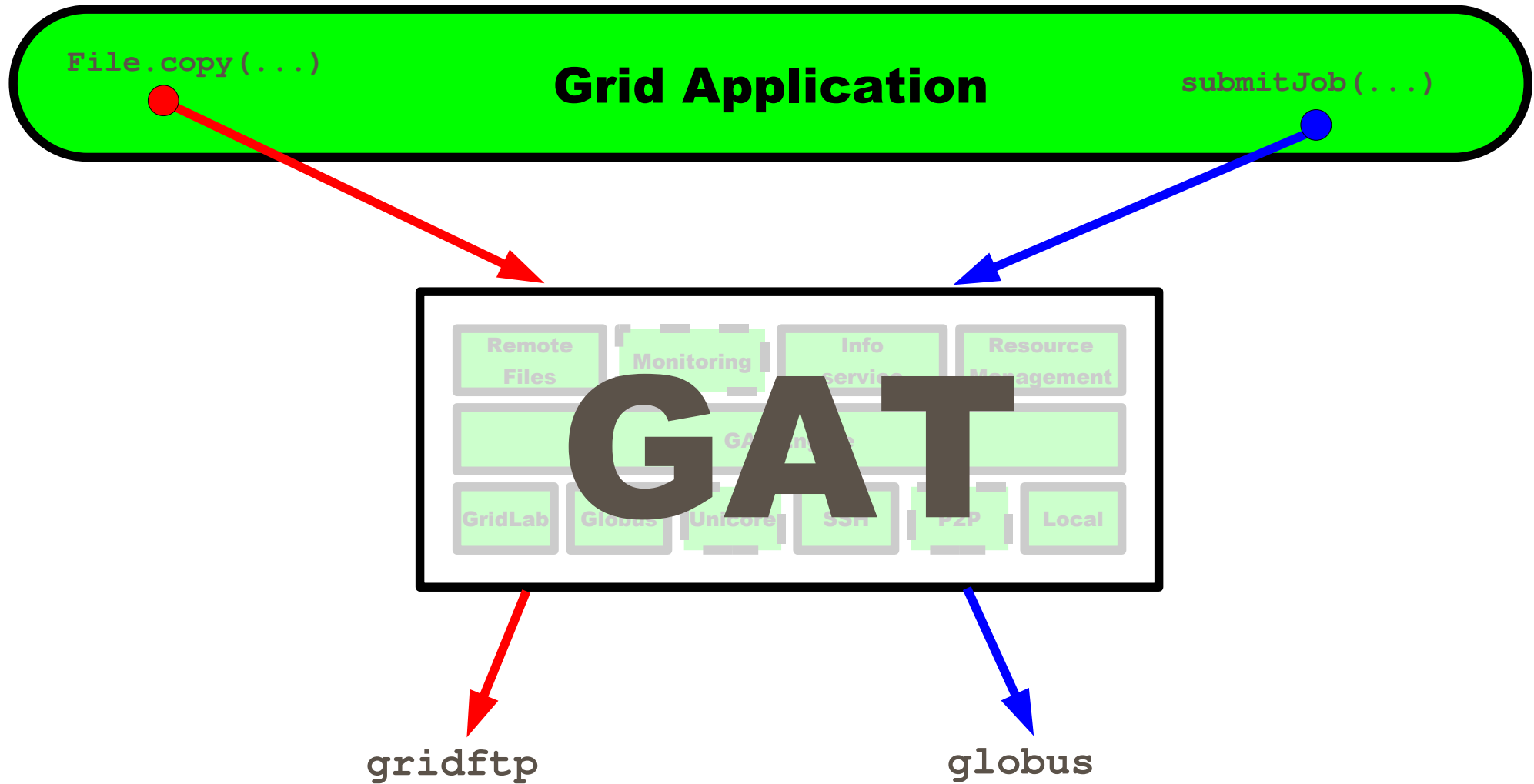
}
catch (Exception e)
{
    System.err.println (MessageUtils.toString (e));
}
}
```

- The situation today:
  - Grids: everywhere
  - Grid applications: nowhere
- Why is this?
  - Application programmers accept the Grid as a computing paradigm only very slowly.
- Problems:
  - Interfaces are NOT simple
  - Portability / interoperability
    - Different and evolving interfaces to the 'Grid'
    - Environment changes in many ways
    - WSDL and web services do not solve all these problems

- GAT: Grid Application Toolkit
  - API and Toolkit for developing and running portable grid applications independently of the underlying grid infrastructure and available services
- GAT is used by applications to access grid services
- **Simple** API
- GAT Adaptors (“plugins”)
  - Connect GAT to grid services
  - Allow for multiple providers (Globus, Unicore, ProActive, ...)
- GAT does not aim to replace existing “grid infrastructure.”
- Open source, BSD-like licence

- Applications make GAT-API calls for grid operations
  - Applications link against GAT
- Applications run irrespective of available infrastructure
  - GAT Engine loads all available adaptors **at runtime**
  - Upon a call to the GAT-API the GAT Engine determines which adaptor(s) provide the “grid operation”
  - Upon “grid operation” failure another adaptor may be called
- There exist a set of default adaptors which provide default local capabilities
  - Grid applications can thus be compiled, linked, and tested without any available grid services
  - The same application executable can run in a “full grid environment.” **No recompilation / linking**

- Security (deal with passwords, credentials, etc)
- Grid I/O
  - File operations, remote file access, file replication
  - Inter-process communication
- Resource Management
  - Resource brokering
  - Forking grid applications, job management
- Application Information Management
  - Global repository for application specific information
  - Query this information repository
- Monitoring
  - Grid monitoring
  - Application monitoring and steering





# ***File Copy with GAT (C++)***

---

```
#include <GAT++.hpp>

void RemoteFile::GetFile (
    GAT::Context context, std::string source_url,
    std::string target_url) throws GAT::Exception {
    GAT::File file (context, source_url);
    file.Copy      (target_url);
}
```

# ***File Copy with GAT (C++)***

---

```
#include <GAT++.hpp>

void RemoteFile::GetFile (
    GAT::Context context, std::string source_url,
    std::string target_url) throws GAT::Exception {

    GAT::File file (context, source_url);
    file.Copy      (target_url);
}
```

- What is GAT and why do we need it?
- **JavaGAT overview and structure**
- Security
- Grid I/O

## Grid Application

**Files**

**Monitoring**

**Info  
service**

**Resource  
Management**

## GAT Engine

**GridLab**

**Globus**

**Unicore**

**SSH**

**P2P**

**Local**

**Legend:**

**Java**

**Done**

**W.I.P**

## Grid Application

API

Files

Monitoring

Info  
service

Resource  
Management

## GAT Engine

GridLab

Globus

Unicore

SSH

P2P

Local

**Legend:**

Java

Done

W.I.P

## Grid Application

API

Files

Monitoring

Info  
service

Resource  
Management

## GAT Engine

ADAPT.

GridLab

Globus

Unicore

SSH

P2P

Local

**Legend:**

Java

Done

W.I.P

`File.copy(...)`

## Grid Application

API

Files

Monitoring

Info  
service

Resource  
Management

## GAT Engine

ADAPT.

GridLab

Globus

Unicore

SSH

P2P

Local

**Legend:**

Java

Done

W.I.P

`File.copy(...)`

## Grid Application

Files

Monitoring

Info  
service

Resource  
Management

## GAT Engine

GridLab

Globus

Unicore

SSH

P2P

Local

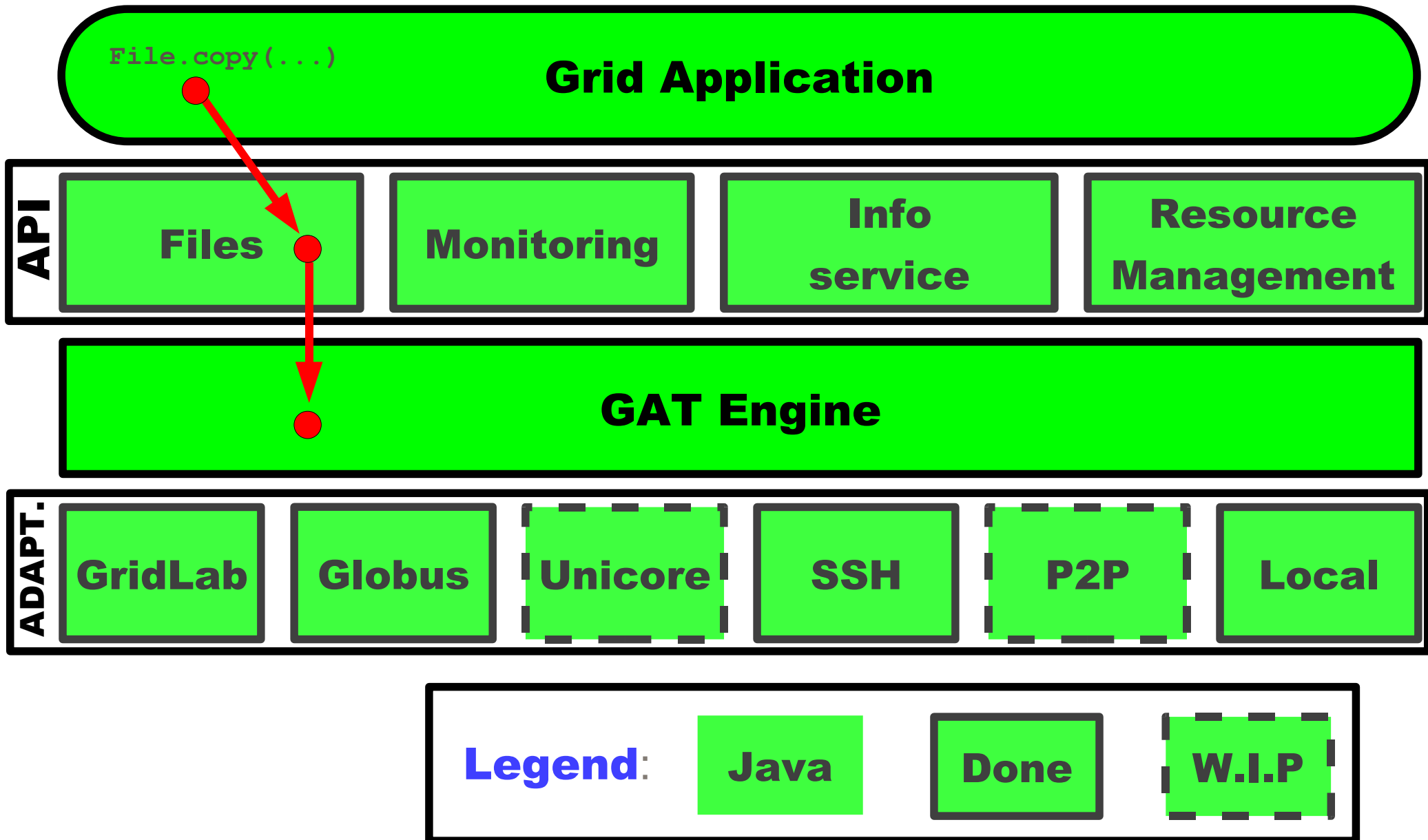
**Legend:**

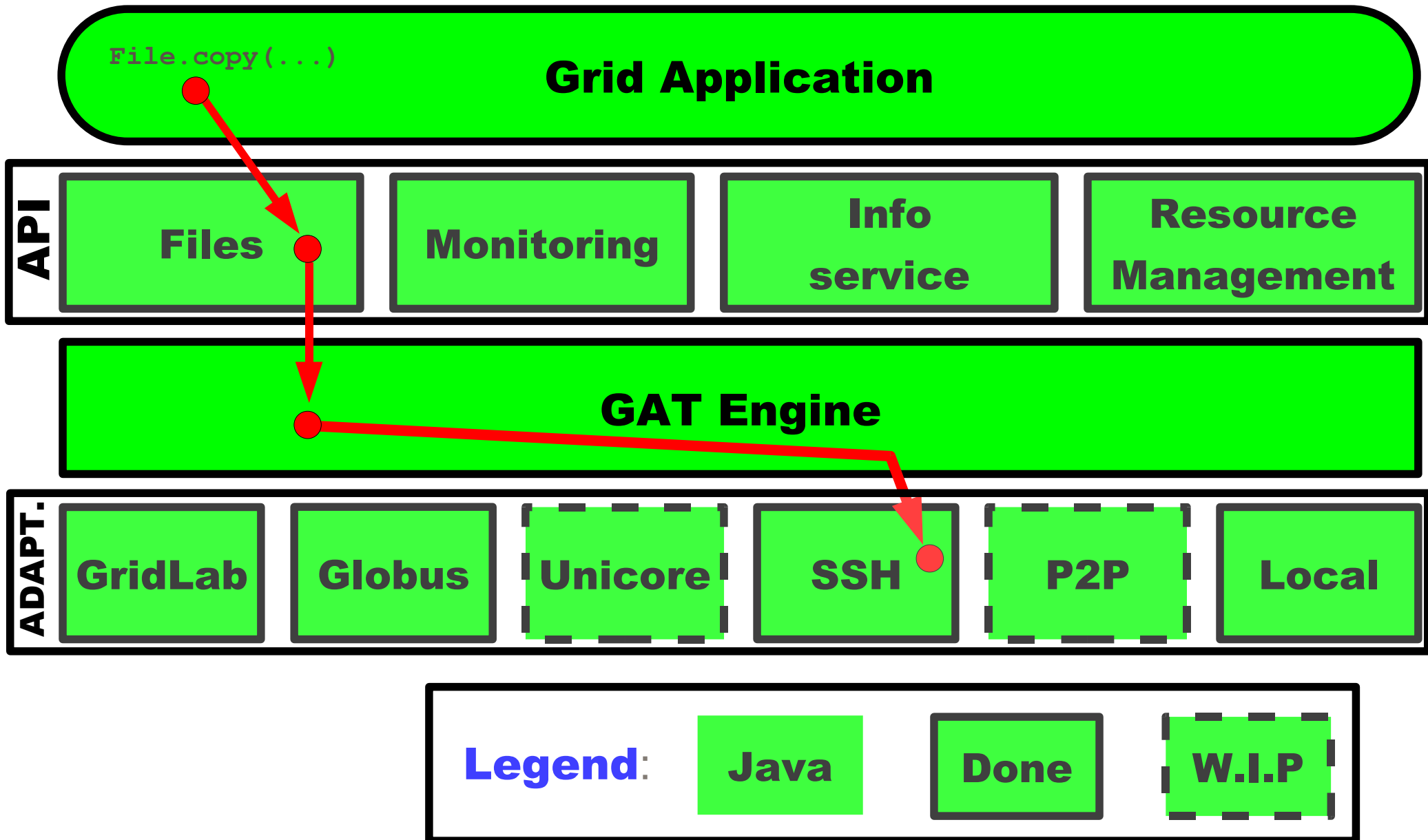
Java

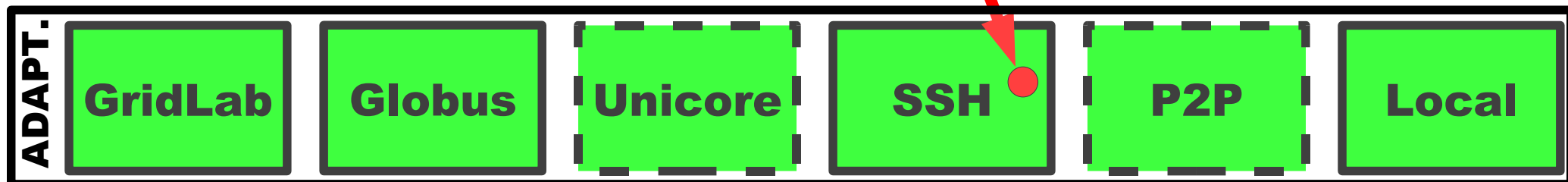
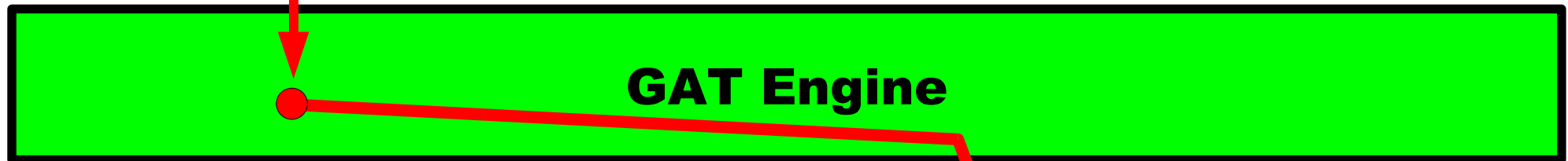
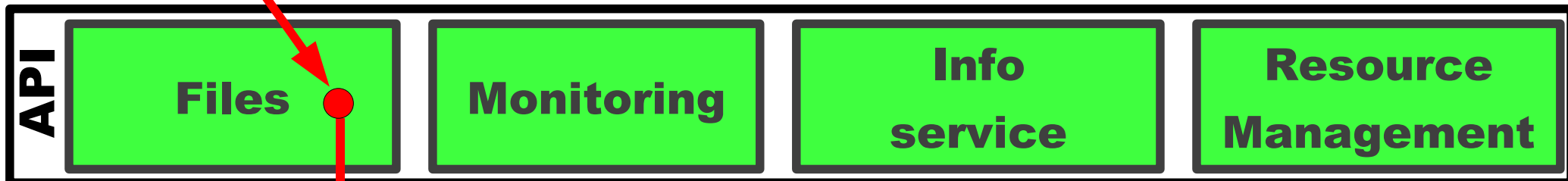
Done

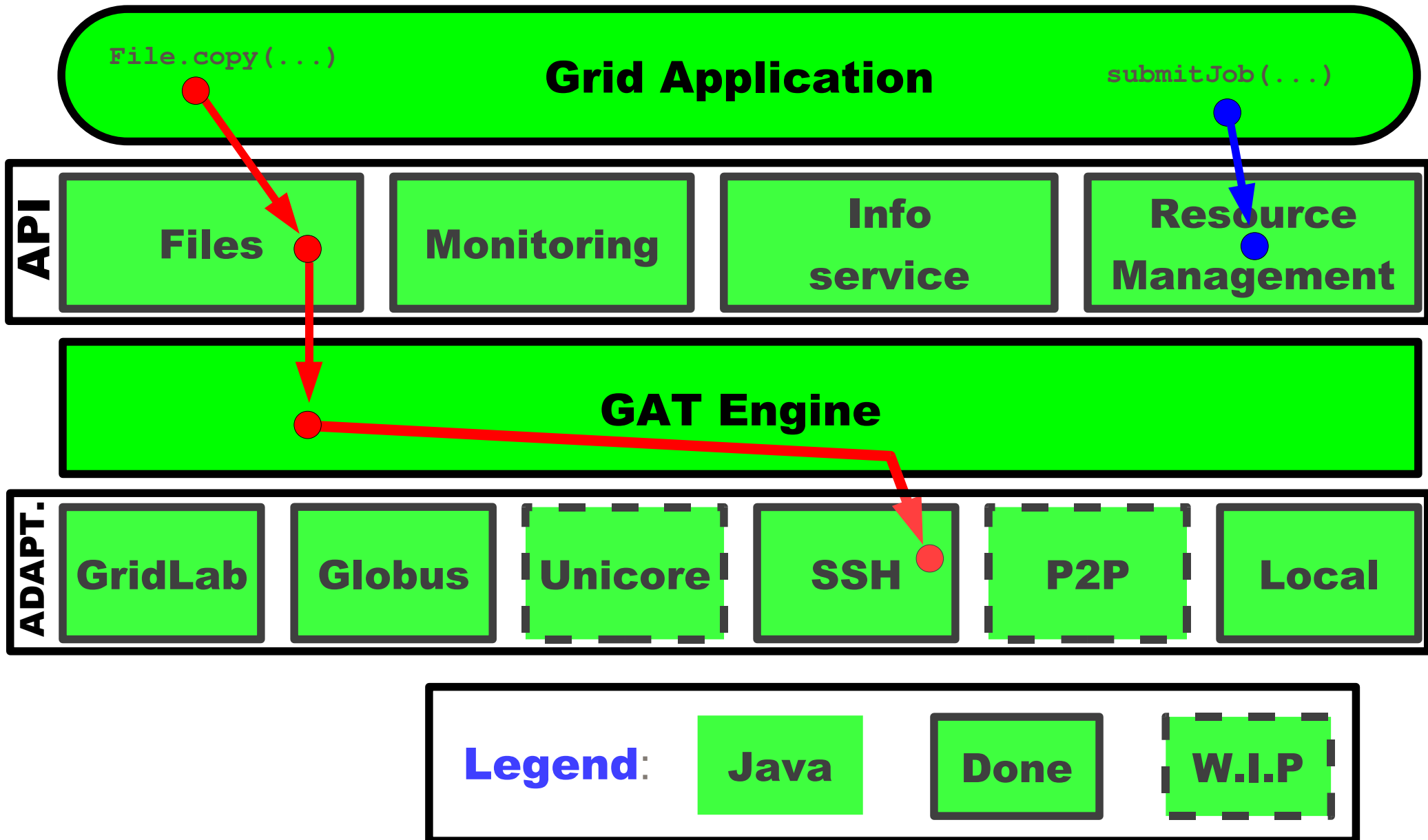
W.I.P

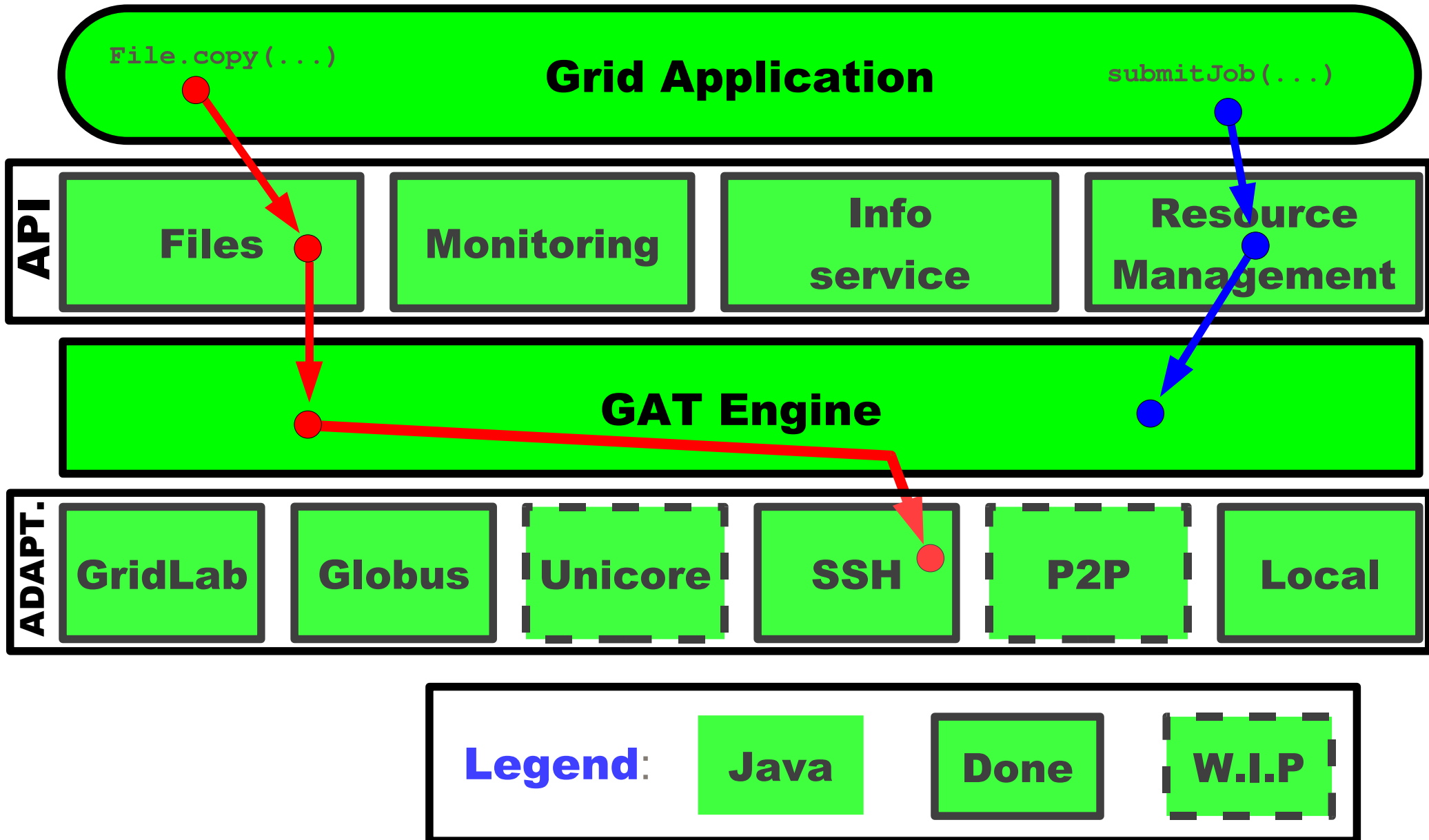


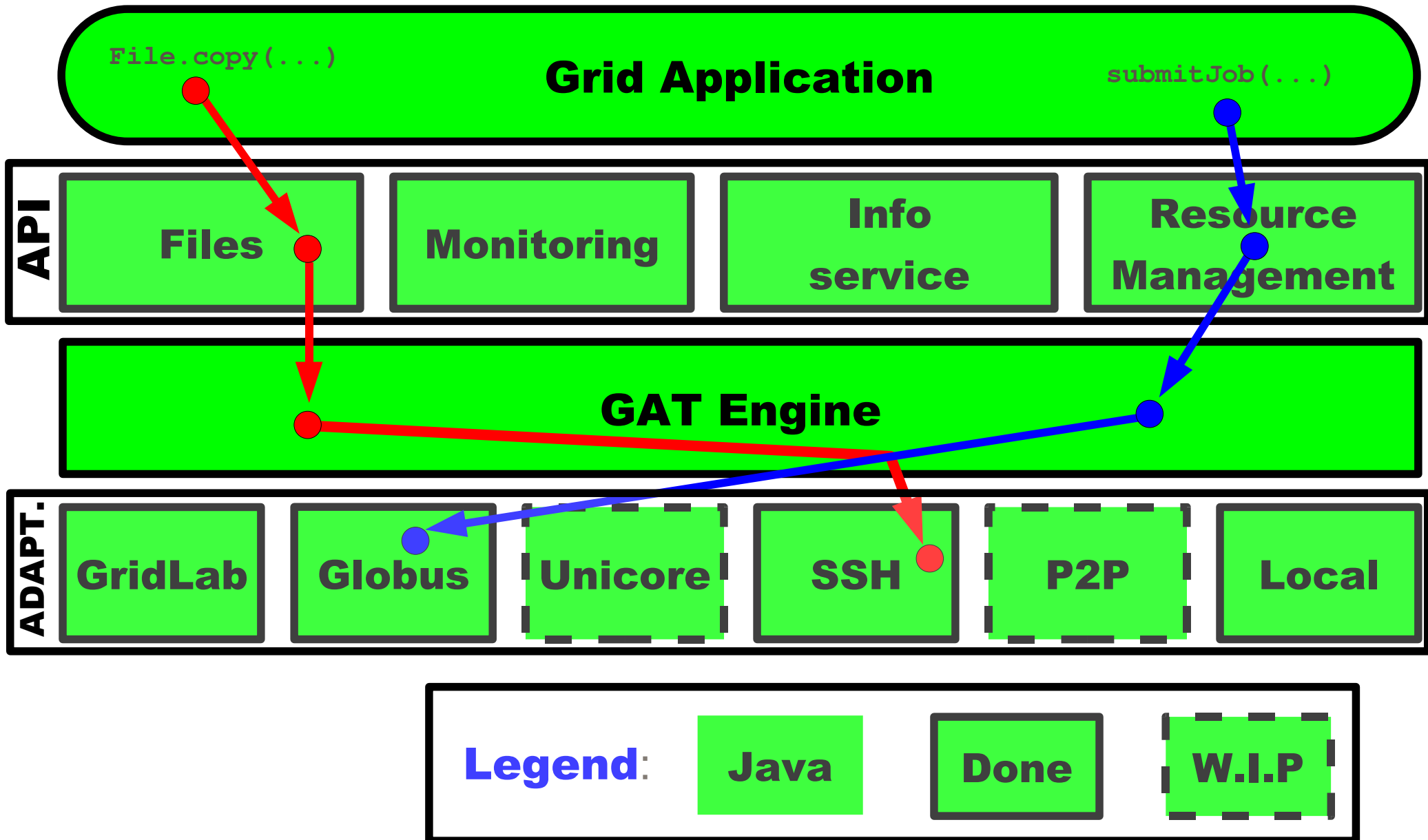


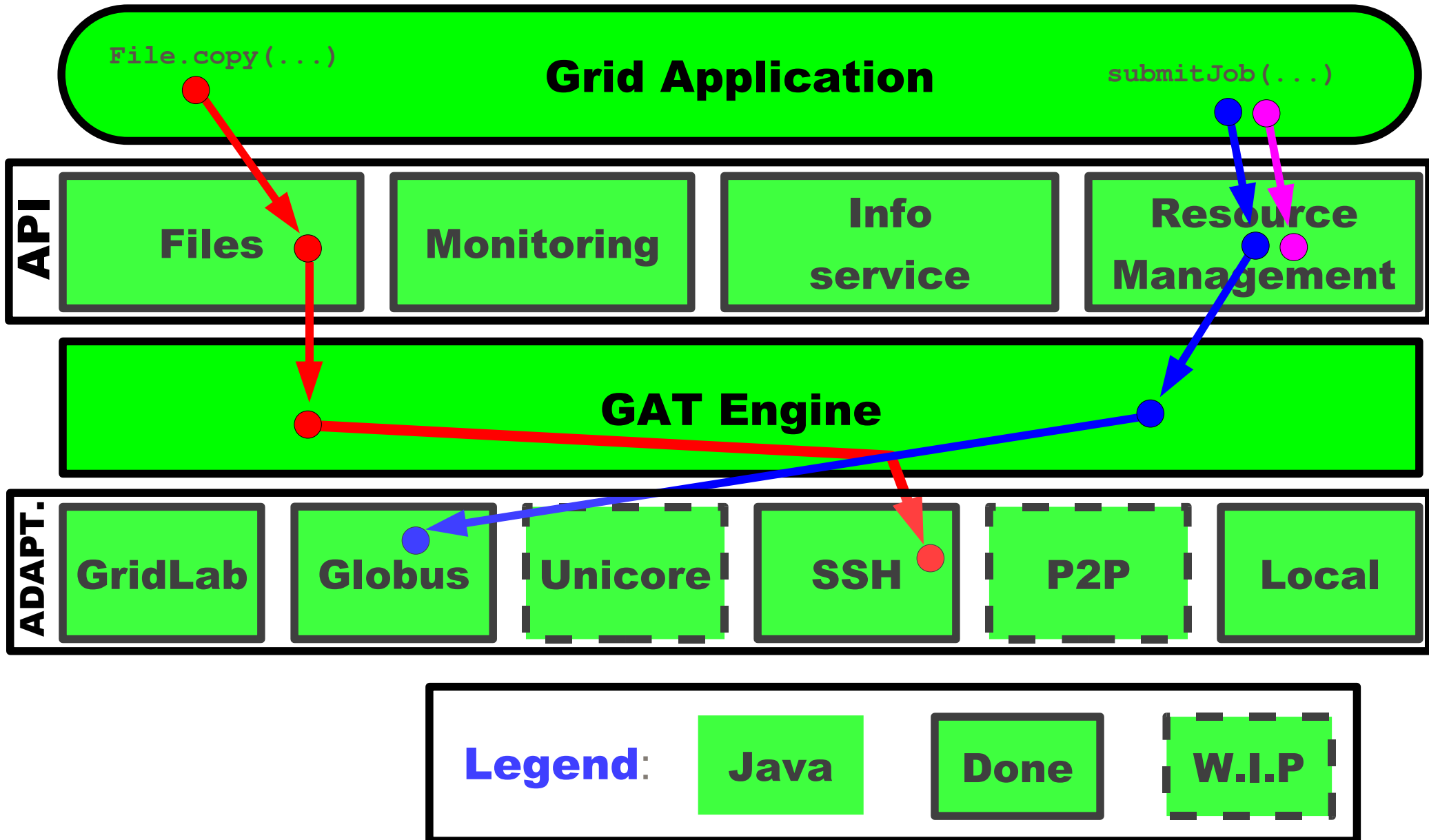


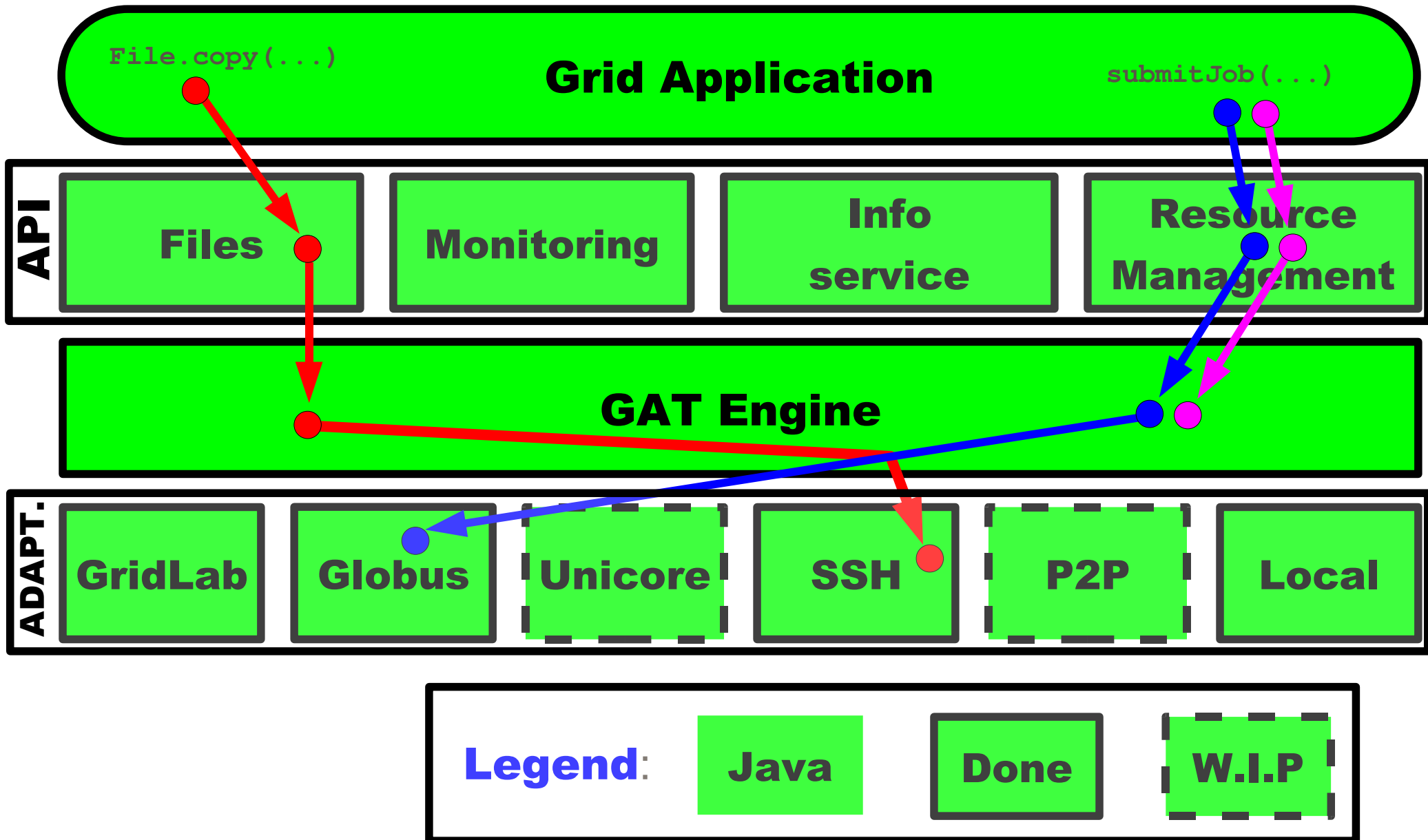




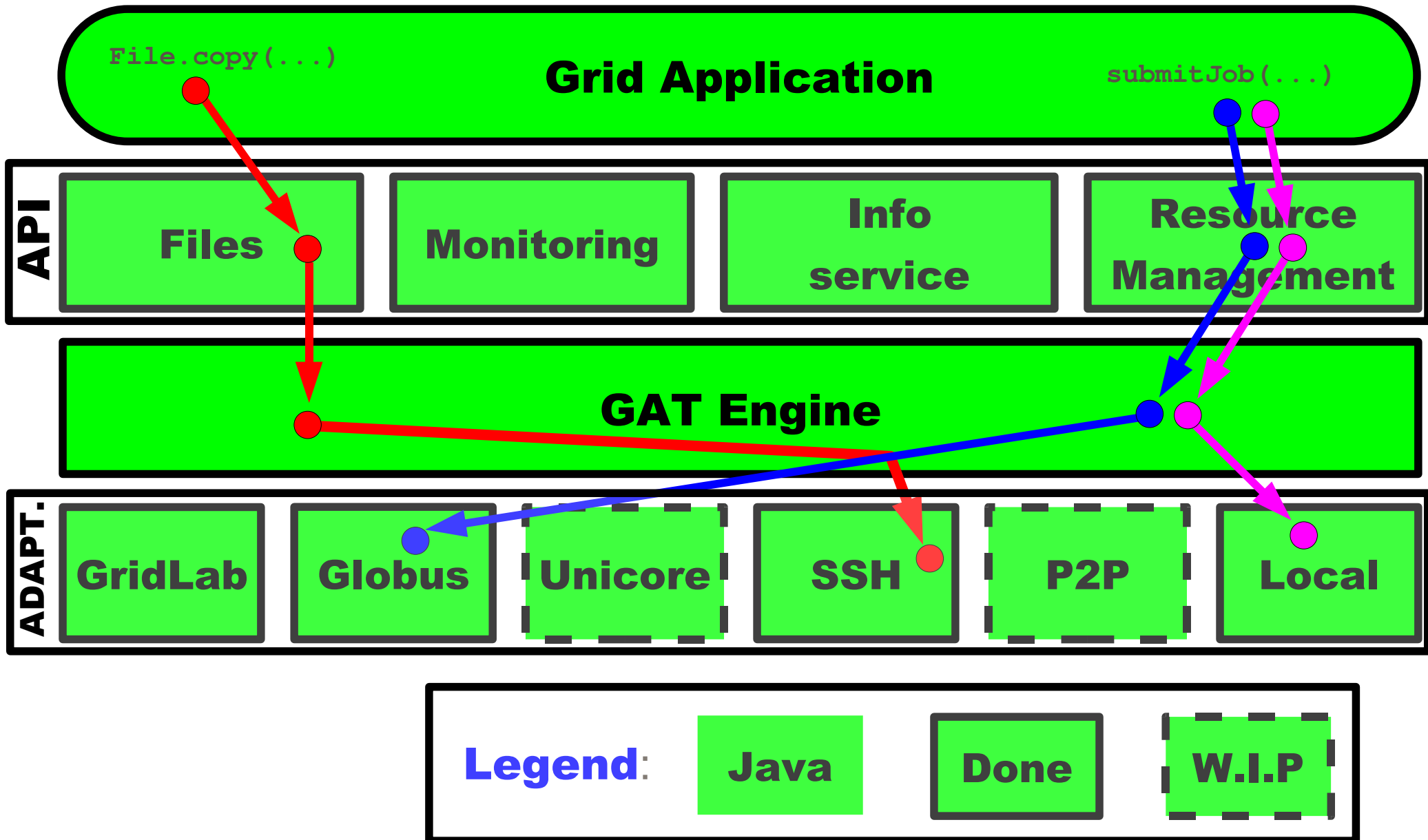












- Adaptors are Java JAR files, dynamically loaded into the application.
- Late binding:
  - The GAT engine selects the best adaptor for each **method**.
  - Example: Create file object.
  - File.copy from site A to site B and C.
  - A -> B copy with GridFTP.
  - A -> C copy with SSH.
- Provides flexibility
- Provides fault tolerance.

- What is GAT and why do we need it?
- JavaGAT structure and overview
- **Security**
- Grid I/O

- Preferences
  - Key value pairs
  - Adaptor-specific instructions
  - Global or local, local overrides global
  - Example: ( "File.adaptor.name", "globus" )
- GATContext
  - Contains security information
  - Contains global preferences
  - There can be more than one context
  - Needed to create GAT objects
- GAT
  - Factory for all GAT objects
- GAT Exceptions
  - Nested, helps debugging (Needed because of late binding)

- URI
  - Slightly different semantics compared to java.net.URI
  - Use the right number of /'s in the URI's
  - Full URI is easy....
    - `protocol://machine/<path>file`
  - ....but some fields may be blank
    - `file:///output` (local file in current directory)
    - `file:///output` (local file in root (/) directory)
    - `file:///tmp/output` (local file in /tmp directory)
    - `ftp://10.0.0.1/output` (remote file in default ftp dir)
- Use the right scheme (protocol) in the URI:
  - JavaGAT can choose (late binding):
    - `any://`
  - Force a specific adaptor (early binding):
    - `ftp://`, `gsiftp://`, `http://`, `file://`, ....

- SecurityContext
  - A container for security Information.
- Abstract, use subclasses
  - PasswordSecurityContext
  - CertificateSecurityContext
  - MyProxyServerCredentialSecurityContext
- Typically not needed if default credentials / private keys are used

```
GATContext context = new GATContext();

SecurityContext pwd =
    new PasswordSecurityContext(username, password);

SecurityContext cert =
    new CertificateSecurityContext(keyfile, username, passphrase);

// add them to the GAT context
context.addSecurityContext(pwd);
context.addSecurityContext(cert);
```

- What is GAT and why do we need it?
- JavaGAT structure and overview
- Security
- **Grid I/O**



# ***Grid I/O Use Cases***

---

- Copy, move, read, write files on the grid
- Random access to remote files
- Replicate files between different grid sites
- Inter-process communication

- File
  - FileInputStream / FileOutputStream
  - RandomAccessFile
- } Extend java.io classes
- ↓
- Grid-enable your code, just by replacing one "new" statement
- LogicalFile
    - Replicated file support
  - Basic Inter-process communication
    - Endpoint
    - Pipe
    - PipeListener

- Represents both files and directories (like java.io)
  - canRead, canWrite
  - delete
  - mkdir
  - list
  - **copy**
  - **move**
  - ...



# ***GAT File example***



```
package tutorial;

import org.gridlab.gat.*;
import org.gridlab.gat.io.File;

public class RemoteCopy {
    public static void main(String[] args) throws Exception {
        GATContext context = new GATContext();

        URI src = new URI(args[0]);
        URI dest = new URI(args[1]);
        File file = GAT.createFile(context, src);           // create file object

        file.copy(dest);                                   // and copy it
        GAT.end();
    }
}
```



# ***GAT File Streaming Example***

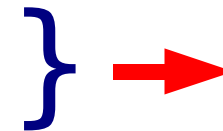


```
package tutorial;

class RemoteCat {
    public static void main(String[] args) throws Exception {
        GATContext context = new GATContext();
        URI loc = new URI(args[0]);

        FileInputStream in = GAT.createFileInputStream(context, loc);
        InputStreamReader reader = new InputStreamReader(in);
        BufferedReader buf = new BufferedReader(reader);

        while(true) {
            String result = buf.readLine();
            if(result == null) break;
            System.out.println(result);
        }
        in.close();
        GAT.end();
    }
}
```



Standard  
java.io  
classes

- GAT will be supported for the foreseeable future
- GAT is being standardized within GGF
  - This will take time
  - Standard is called SAGA (Simple API for Grid Applications)
- Java reference implementation of SAGA is being developed by us
  - A layer on top of the JavaGAT
- API differences
  - Conceptually, SAGA is very close to GAT
  - SAGA adds task model to GAT for asynchronous grid operations
  - SAGA is Posix-like, not Java-like
  - SAGA is less configurable by design (e.g., no preferences)

- Download is anonymous, so we don't know
- Max Planck Institute for Astrophysics in Garching
- D-Grid
- Astrogrid
- Louisiane State University
- University of Texas
- AMOLF, Institute for Atomic and Molecular Physics
- The Dutch Virtual Labs for E-science project (VI-e)
- The workflow system Triana (University of Cardiff)
- Georgia State University
- Vrije Universiteit Amsterdam (Ibis, teaching)
- The Multimedial project
- Zuse Institute Berlin, Germany
- VU Medical Center Amsterdam

- The GAT provides a simple and stable API to various Grid environments
- But powerful!
- Independent of grid middleware
- Portable
- Downloads:
  - [www.cs.vu.nl/ibis](http://www.cs.vu.nl/ibis)
  - Distributions
  - Anonymous SVN access at [gforge.cs.vu.nl](http://gforge.cs.vu.nl)
  - Java Platforms: any (Java 1.5 or higher)
  - Support via gforge site mail, forum, bug tracking

JavaGAT 2.0  
release  
candidate 1  
available

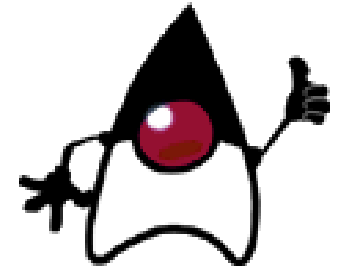


# Why Java?

---



- Java is widely used, object oriented.
- Secure (language, sandbox)
- Java is “write once, run everywhere”.
  - Compile application on your desktop machine.
  - This creates machine independent bytecode.
  - Copy application files and the GAT to any grid site (portal typically does this for you).
  - Just run it. No recompilation / configuration.
- Performance of current JITs is good.
  - Compiled (just-in-time)
  - Runtime, profile-driven optimizations
  - Applications are typically 10% slower than C.
- Ideal for grid computing?



```
GATContext context = new GATContext();  
Preferences prefs = new Preferences();  
prefs.put("File.adaptor.name", "globus");  
context.addPreferences(prefs);
```

```
src = new URI("hello");  
file = GAT.createFile(context, src);
```

OR use local preferences to override globals:

```
file = GAT.createFile(context, morePrefs, src);
```

- SecurityContext
  - A container for security Information.
- Abstract, use subclasses
  - PasswordSecurityContext
  - CertificateSecurityContext
  - MyProxyServerCredentialSecurityContext
- Notes restrict the access to the context
  - Avoid broadcasting of passwords / credentials
  - Restrict access to a set of hosts or adaptors
  - One or more notes -> restricted to those adaptors/hosts
  - No notes -> any adaptor can use context for any host
- Typically not needed if default credentials / private keys are used

# ***GAT Remote Random Access Files***

---

- Random access to remote files
  - read
  - write
  - seek
  - length
  - ...

- LogicalFile class
  - An abstract representation of a set of identical physical files
  - addFile / addURI
  - removeFile / removeURI
  - **replicate(URI destination)**
- Replicate a logical file to a new location.
  - Copy one of the files in the set to the new location
  - Choose the “best” one
    - Closest in terms of bandwidth
    - Cheapest
    - ...
    - Depends on adaptor
- Typically used for staging in files for jobs
  - Resource broker (or GAT) chooses one of the files in the set

- File, streams and random file access:
  - Local files
  - GridLab data service
  - FTP, HTTP, HTTPS
  - GridFTP (Globus)
  - SSH, SFTP
- Logical file (replication):
  - Generic adaptor on top of GAT File
  - Logical file adaptor for GridLab replica service
- Pipe
  - Sockets