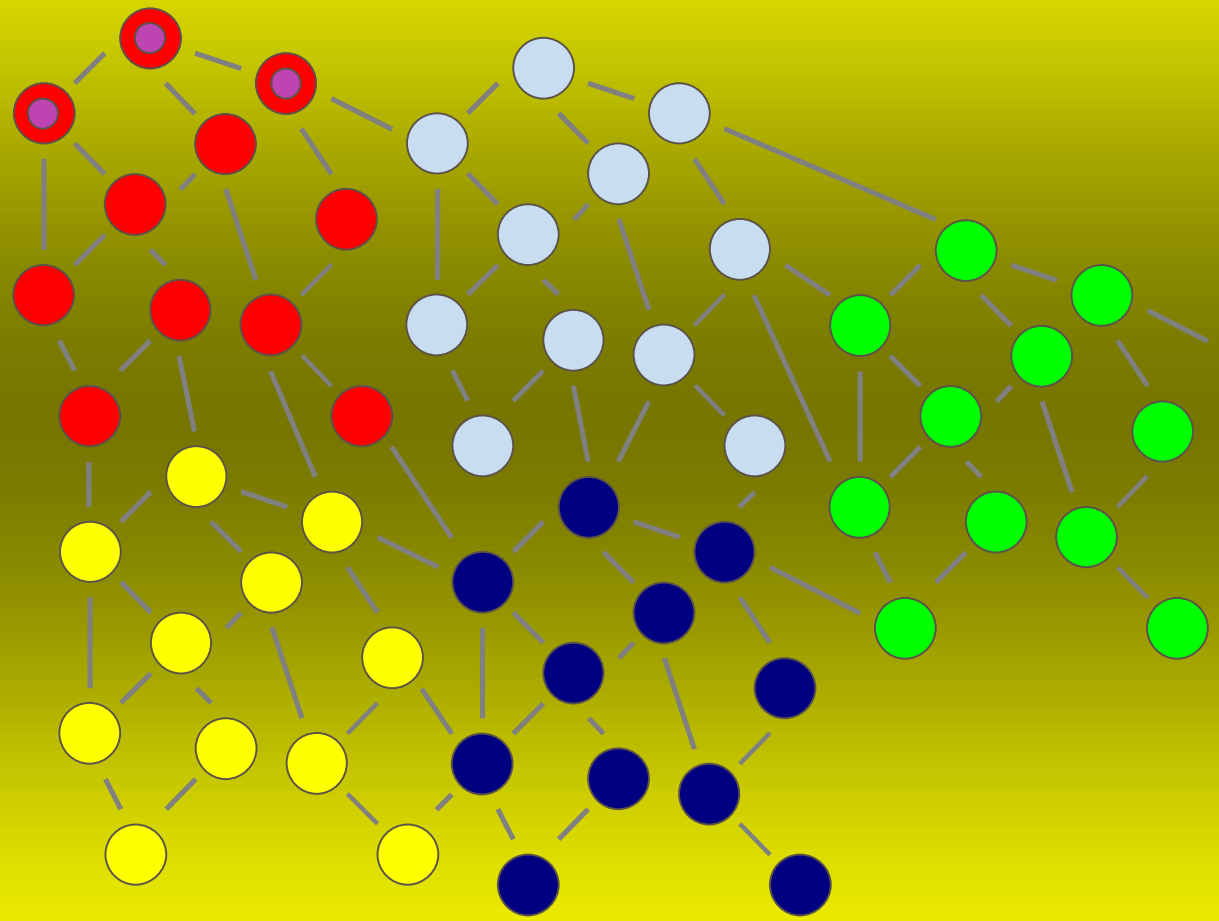
Gridlab: Transparent Grid Programming using the GAT



Ian Taylor, Heidelberg Conference, 29th September 2003

Overview

- Overview of the Gridlab architecture
  - Background
  - Philosophy
  - Architecture
- GAT Uses Case: Triana
  - Overview
  - Software Architecture
  - The use of the GAT
  - Current Implementations
- GAT: Current Status
  - Java and C GAT implementation status

User Scenario: Galaxy Formation

Roger Philp

Cardiff Centre for Computational Science and Engineering

Galaxy Star Formation Simulations

- galaxy and star formation simulations are typically represented by 16-D data sets that require visualization from different perspectives
- a series particles in three dimensions and their associate properties as a snap shots in time.
- user would like to visualize this data as an animation in two dimensions and then vary the point of view and project of that particular two dimensional slice and re-run the animation.

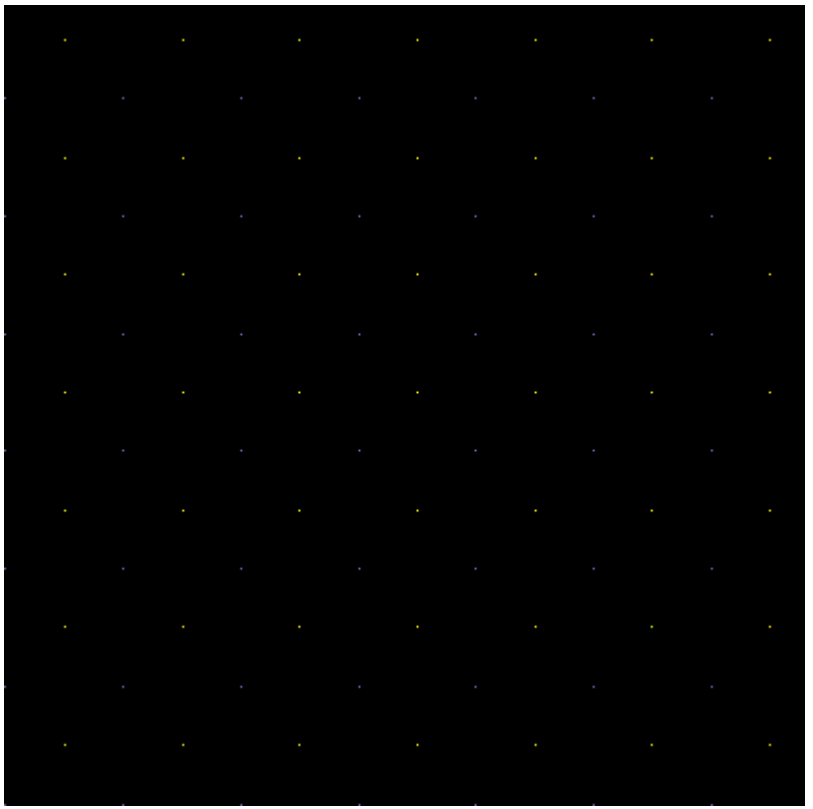
Distribute Galaxy Formation Code ?

GridLab

JAVA

Information Society Technologies

Galaxy Formation Code



GT3 ? Globus ? P2P ?

JXTA ? Unicore ? Jini ?

Grid

Node Node Node Node Node Node

What Does Roger Want ?

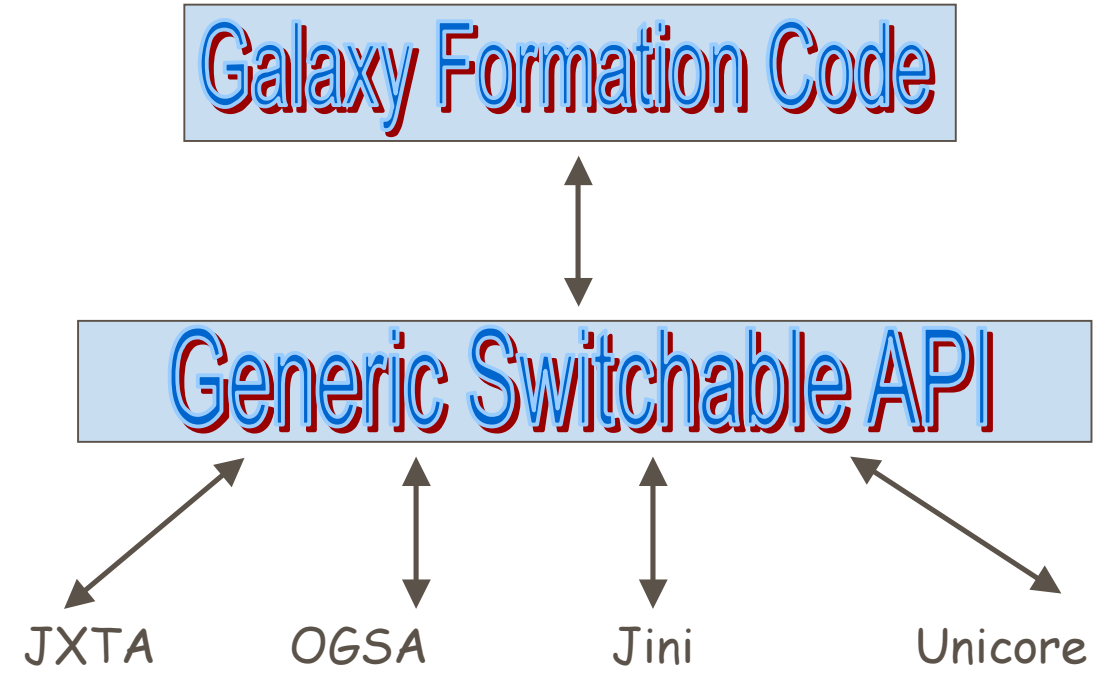
- Not have to learn Globus 2/3?, Jxta, Jini etc.
- Application is >>> more important than mechanism
- Wants it to work in various scenarios
- Compare results from various Grid systems

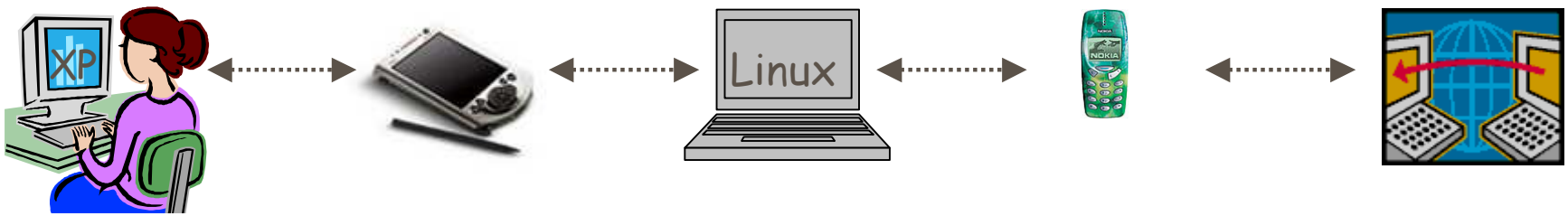
Needs following facilities:

1. Create and run 'Galaxy Code' nodes (Services)
2. Discover 'Galaxy Code' nodes
3. Distribute dataset to nodes
4. Gather results and animate

The minimum number of calls – an afternoon's work……

Roger's Dream .....

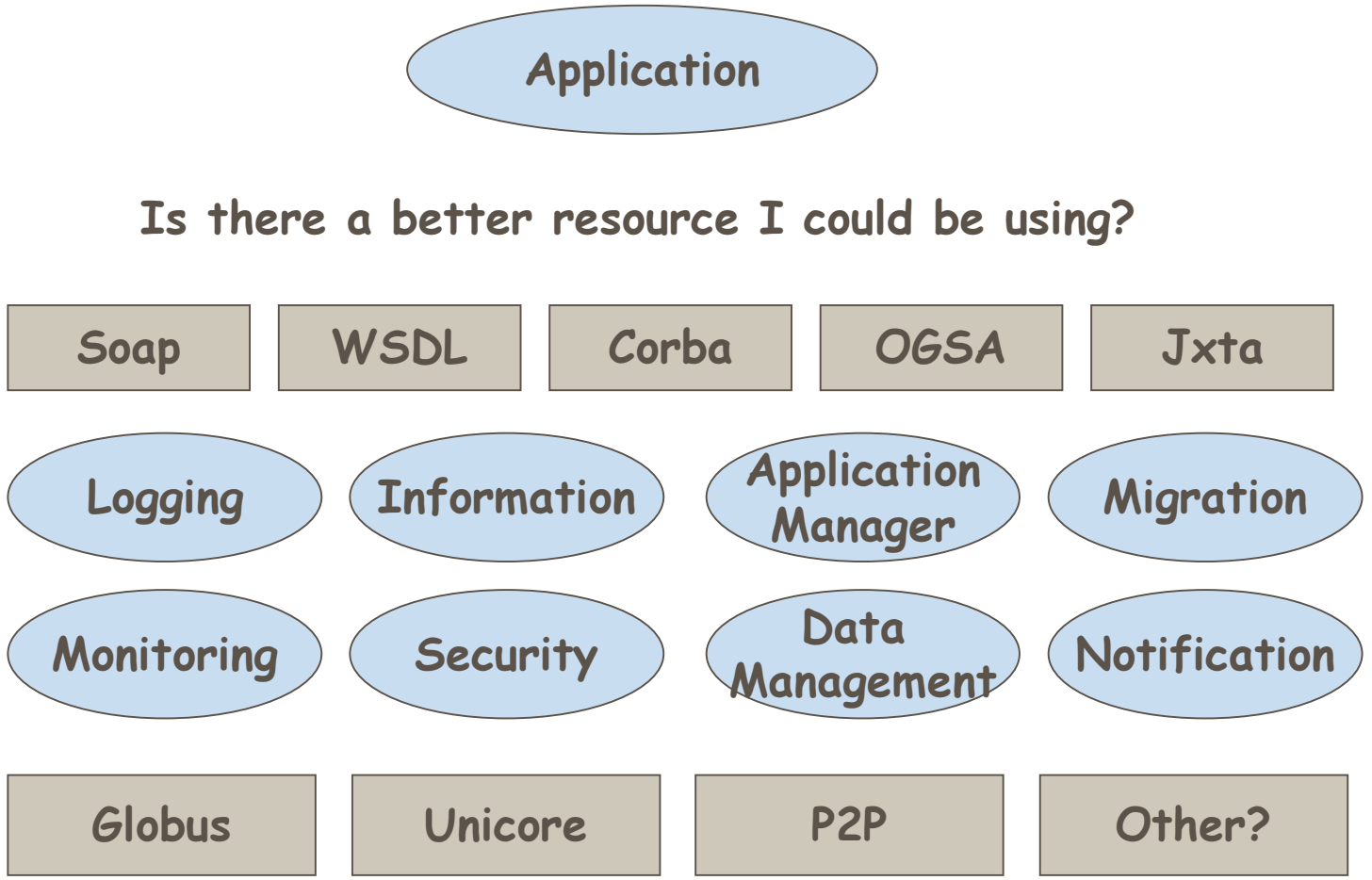




The Gridlab GAT Interface

- Background: An Application Developers Perspective
  - Many middleware choices – evolving standards
  - Which one to choose ?
  - How do I prototype now ??
- Gridlab GAT
  - Provides an insulation layer for applications
  - Contains common calls e.g. submit_job, file_copy etc
  - loosely coupled, dynamically late bound modules
- Application Developers
  - Can prototype and test new application scenarios which make adaptive, dynamic, wild, and futuristic uses of resources

Background: GAT



Heterogeneous Computing Environments

Laptop

Application

GAT

No Network !

Supercomputer

Application

GAT

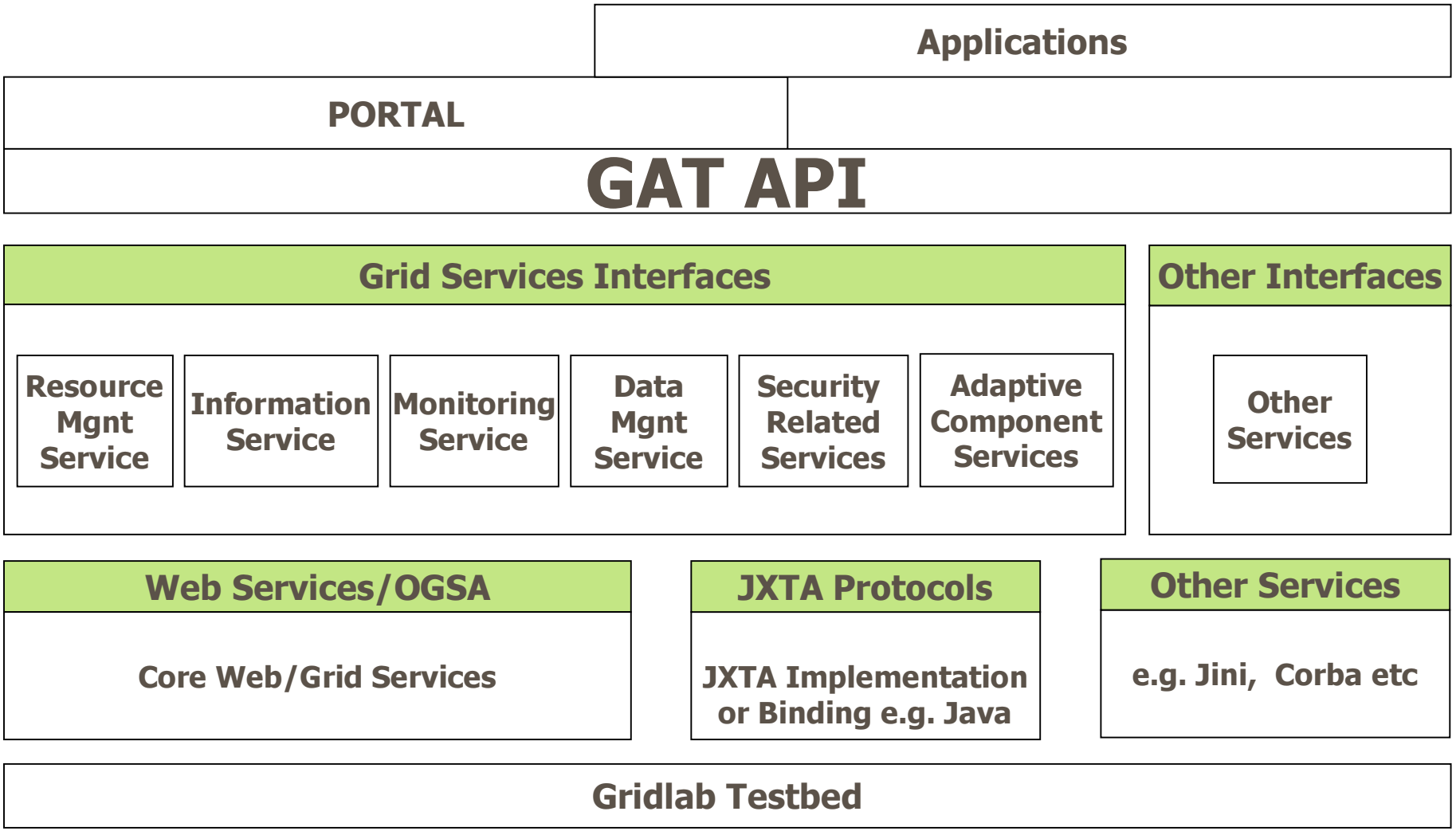
Firewall issues ..

The Grid

Application

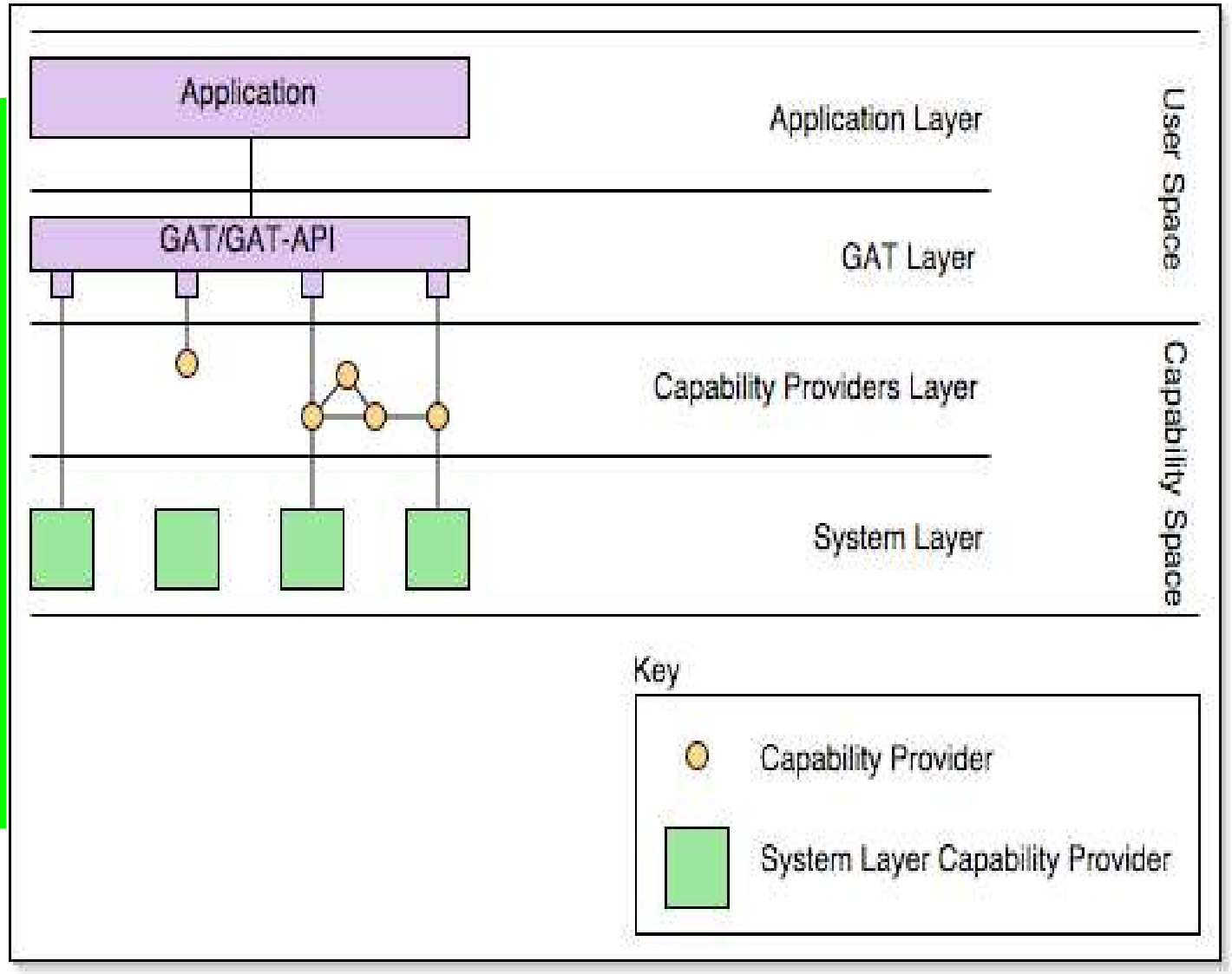
GAT

GAT Overview



GAT Architecture

The GAT framework software architecture is a layered architecture so as to allow for a loose coupling of various software components. In particular the GAT framework software architecture consists of four software layers, e.g. four logical groupings of software components



GAT API: Design

GAT has had input from 2 generic applications and a number of scenarios:

- Triana: www.trianacode.org a generic graphical problem solving environment – detail later
- Cactus www.cactuscode.org:
  - open source problem-solving environment
  - designed for scientists and engineers
  - modular structure easily enables parallel computation across different architectures and collaborative code development between different groups.
  - originated in the academic research community, where it was developed and used over many years by a large international collaboration of physicists and computational scientists
  - Used for black hole simulations

GAT: Further Information

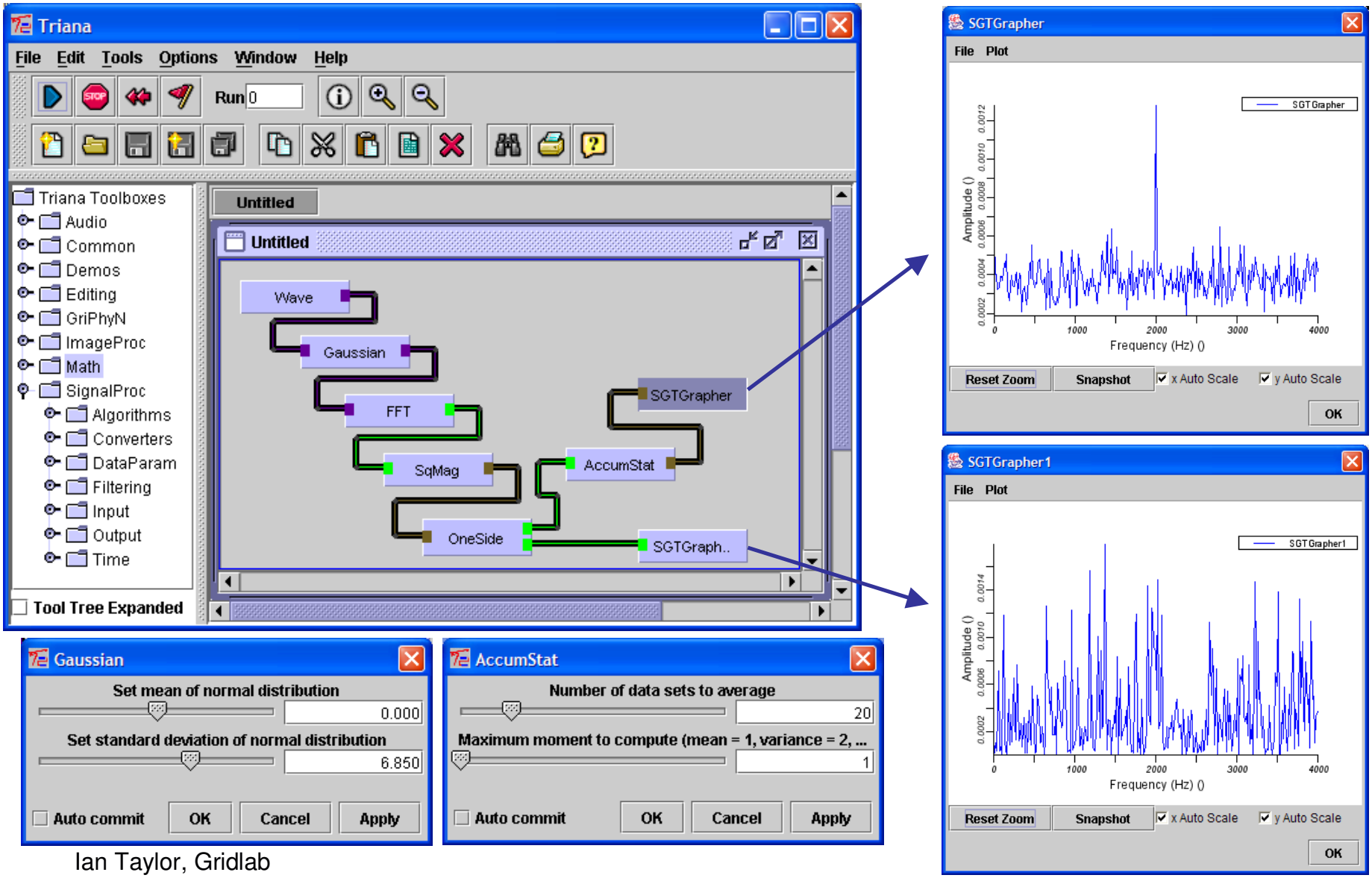
- Two API specification Documents:
  - Non object-based
    - http: //www.gridlab.org/WorkPackages/wp-1/Documents/
    - Gridlab-1-GAS-0003.APISpecification.pdf
  - Object-based
    - http: //www.gridlab.org/WorkPackages/wp-1/Documents/
    - Gridlab-1-GAS-0004.ObjectBasedAPISpecification.pdf

- Mailing List
  - gat@gridlab.org
  - Open subscription

Triana: Application Example

- Overview of Triana
  - Description
  - Distributed Design: with GAT in mind
- GAT Integration
  - Architecture
- Interoperability
  - Swappable components
  - Services e.g. OGSA, Web Services, JXTA etc

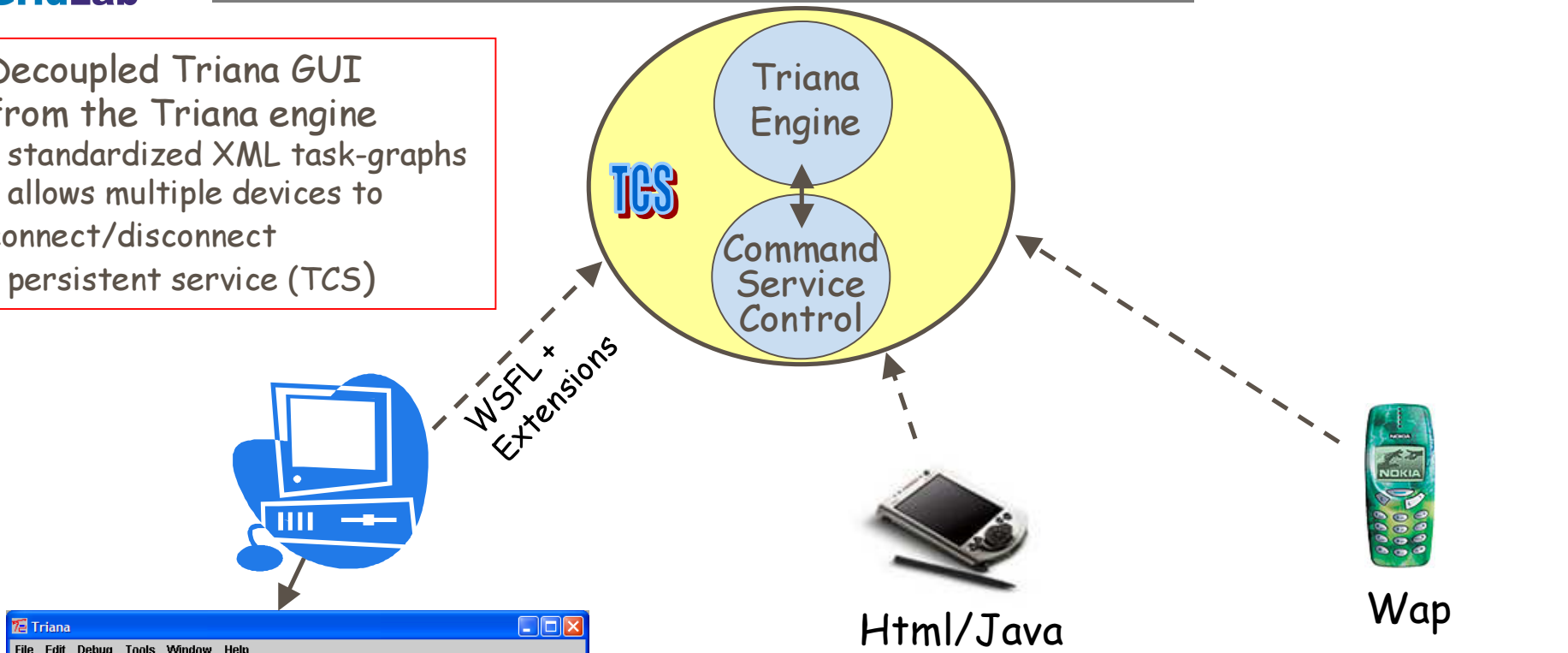
What is Triana?



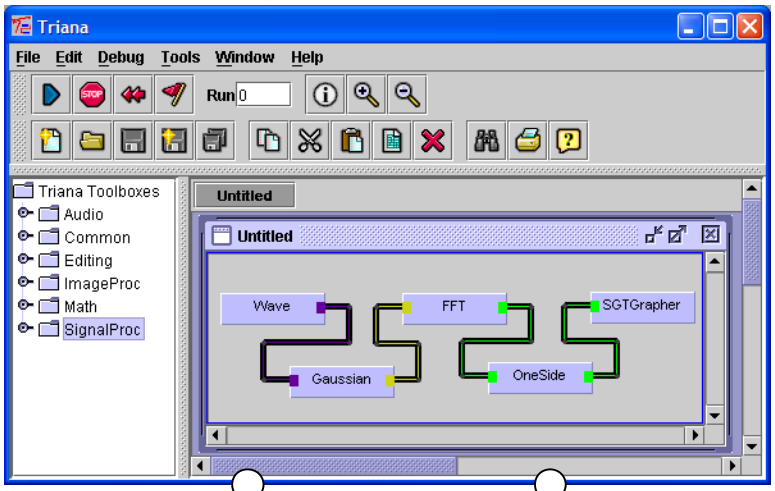
Ian Taylor, Gridlab

Remote Control

Decoupled Triana GUI
from the Triana engine
- standardized XML task-graphs
- allows multiple devices to connect/disconnect
- persistent service (TCS)



Distributed Work-flow



Workflow, e.g. BPEL4WS

Action Commands

Network

Triana Controlling Service (TCS)

Other Engine

Triana Engine

Triana Gateway

Triana Service & Engine

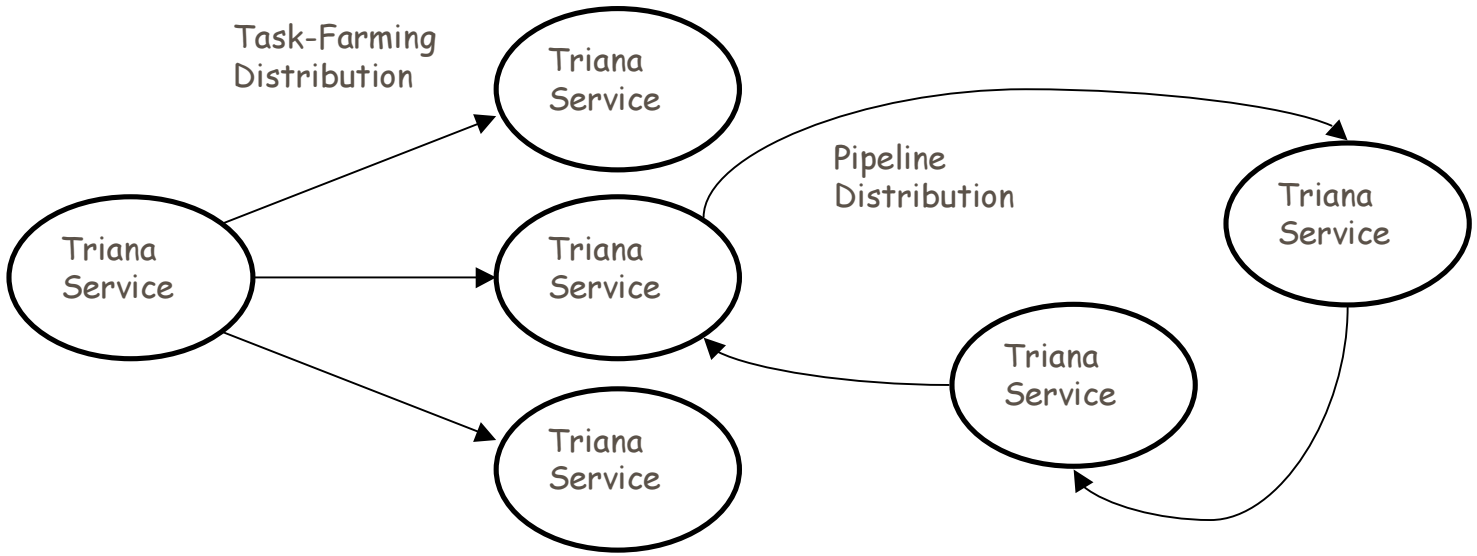
Triana Service & Engine

Triana Service & Engine

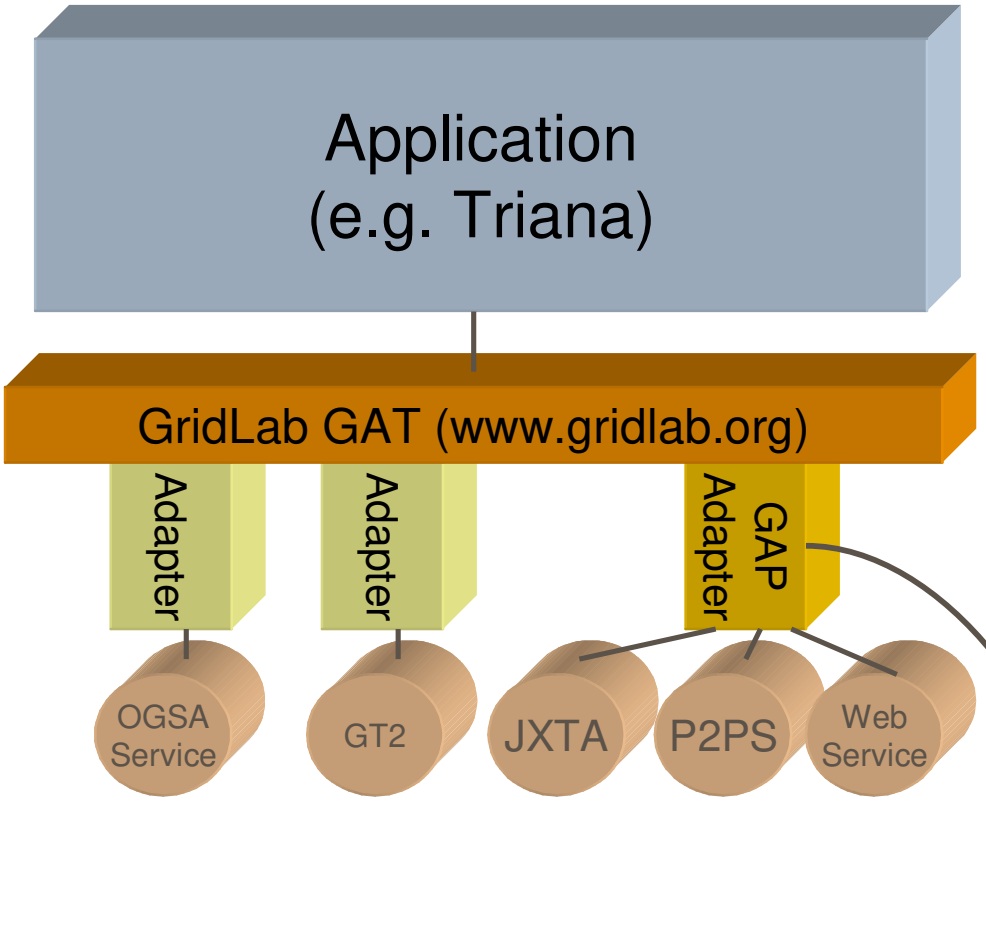
Distributed Triana Work-flow
- flexible distribution: based around Triana Groups
- HPC and P2P distribution

Triana Prototype

- Distributed Triana Prototype
  - Based around Triana Groups i.e. aggregate tools
  - Distribution policies:
    - HTC - high throughput/task farming
    - Peer to Peer - allow node to node communication



Triana GAT Architecture

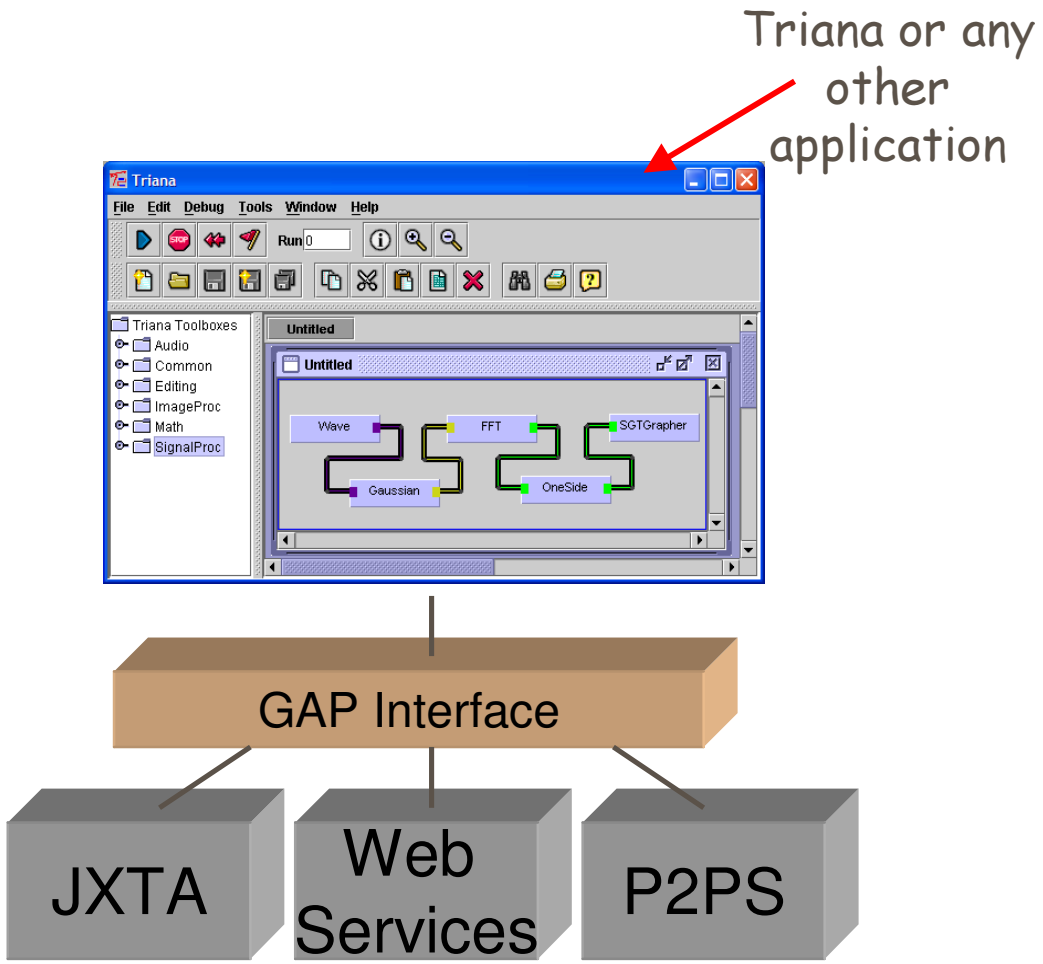


GAP is our P2P Application-level Interface

Triana and the GAP Interface

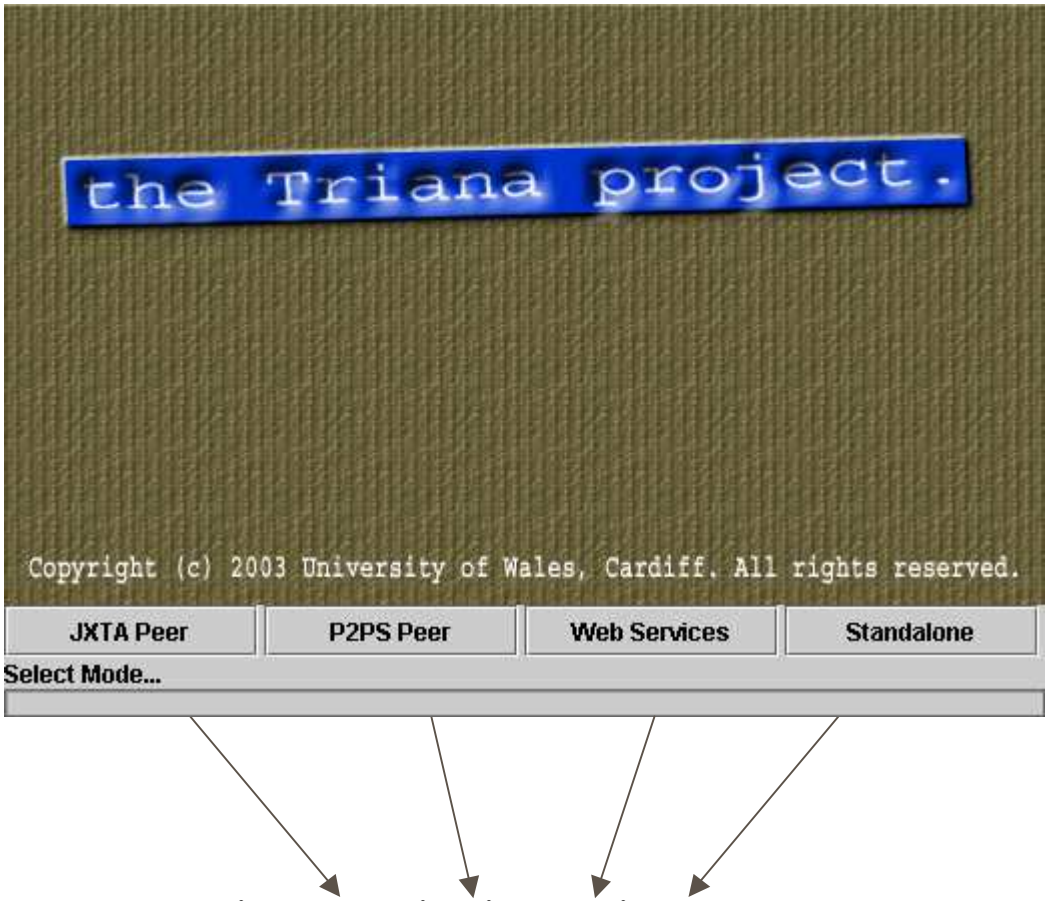
- Interface between Application and Peer-to-Peer Middleware
  - Provides an insulation layer for P2P applications
- loosely coupled, dynamically late bound modules

- Simple
- Generic
  - Not Triana Specific
  - Contains common calls e.g. advertise_service, discover_service, create_pipe etc



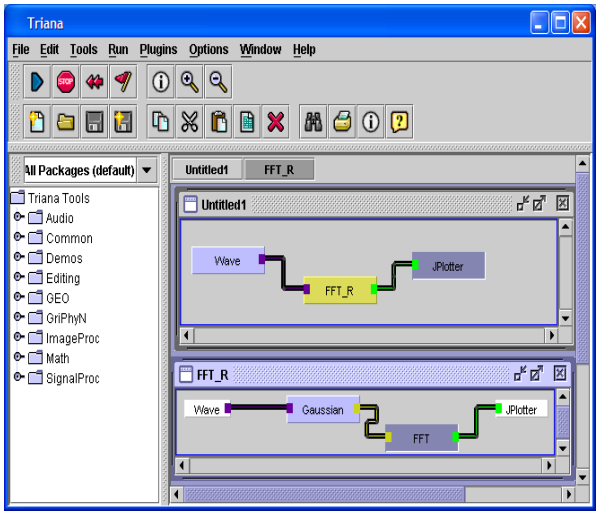
For more info on Triana Distributed Implementation, see www.trianacode.org

A User's Perspective

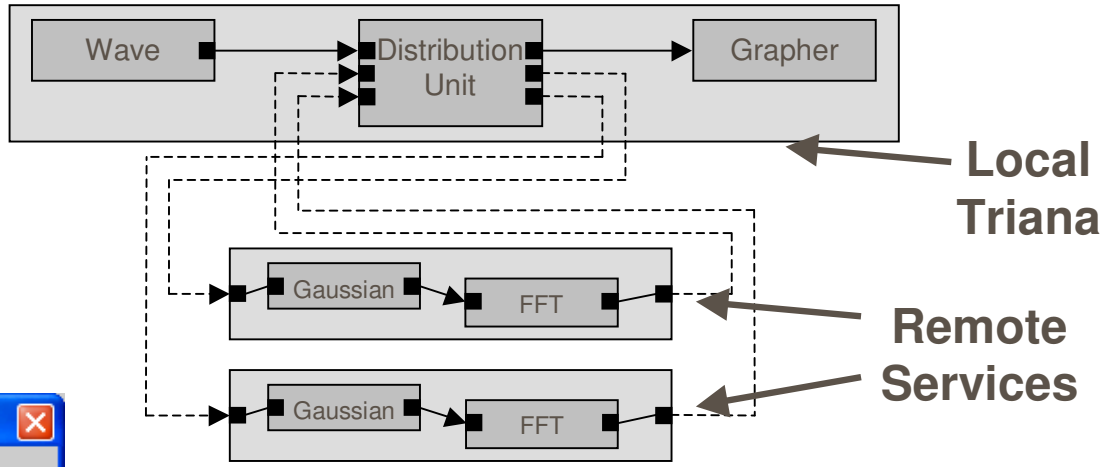


Flash screen: choose which Binding you want to run within

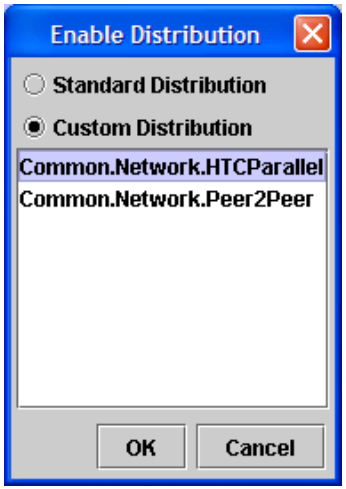
Custom Distribution



- The workflow is cloned/split/rewired to achieve the required distribution topology



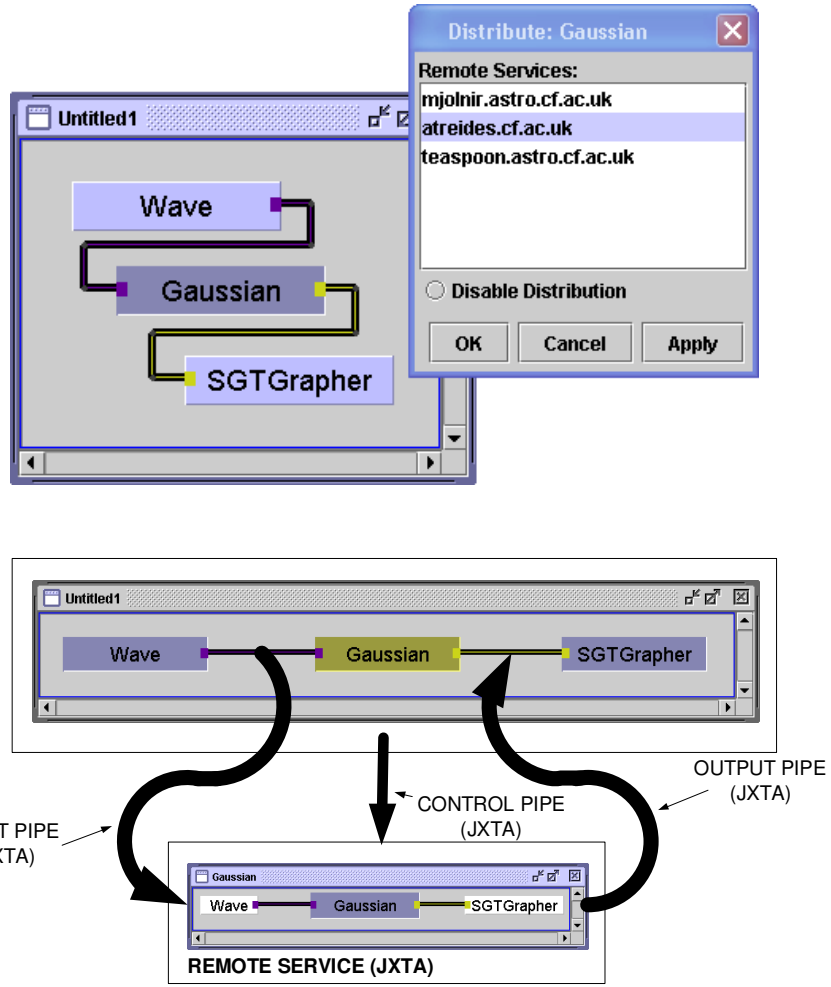
- Custom distribution units allow sub-workflows to be distributed in parallel or pipelined



- Distribution units are standard Triana tools, enabling users to create their own custom distributions

Remote Deployment

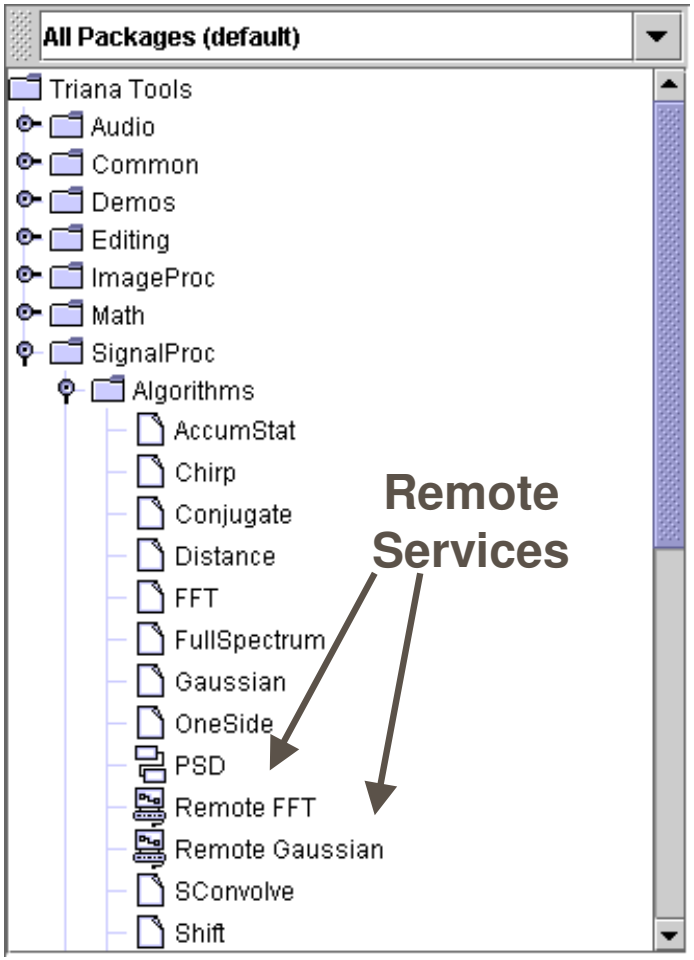
- User can distribute any task or group of tasks (sub-workflow)
- Using the GAP Interface, Triana automatically launches a remote service providing that sub-workflow.
- Input, Output and Control Pipes are connected using the current GAP binding (e.g. JXTA Pipes)



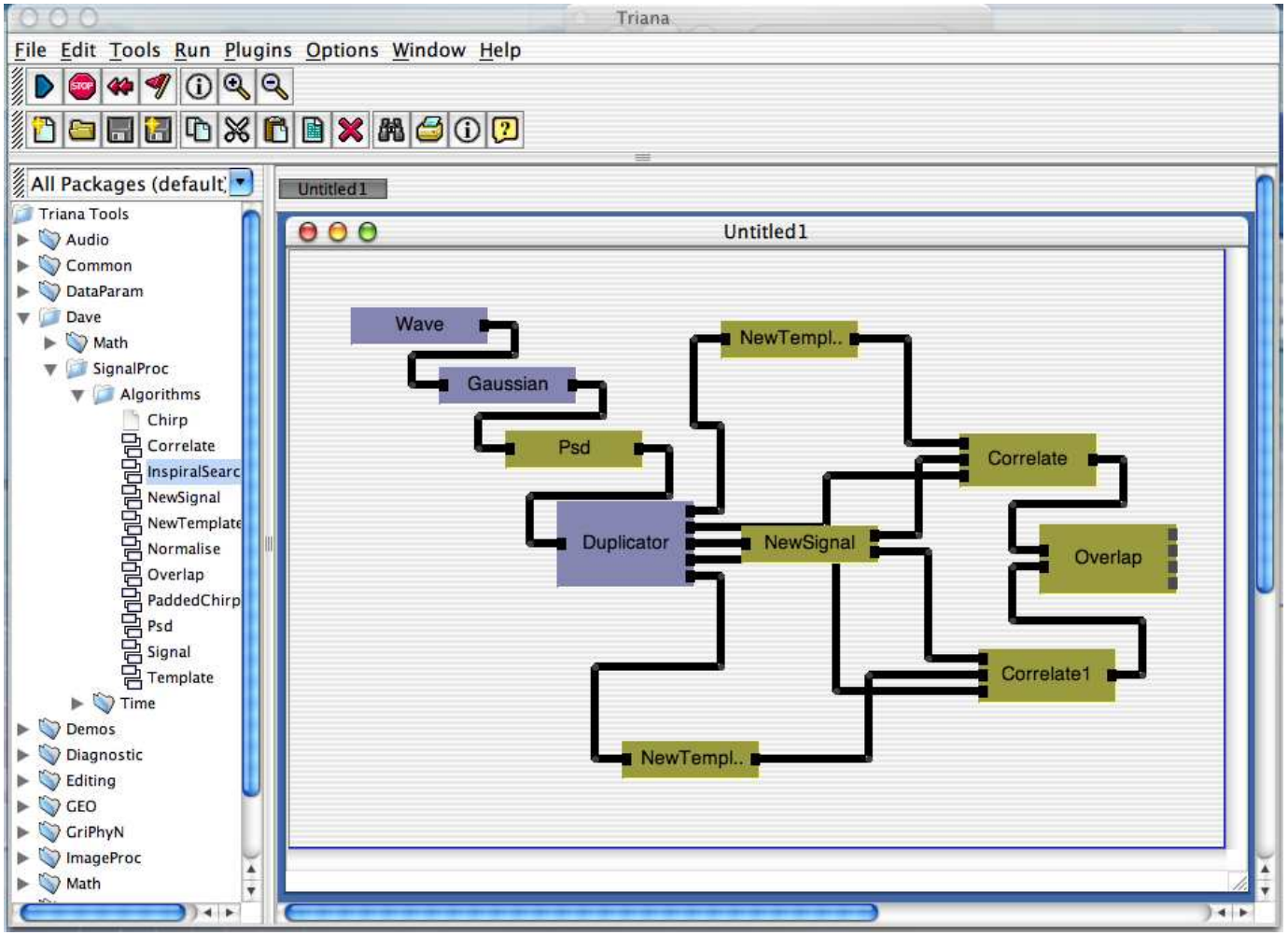
Deploying and Connecting To Remote Services

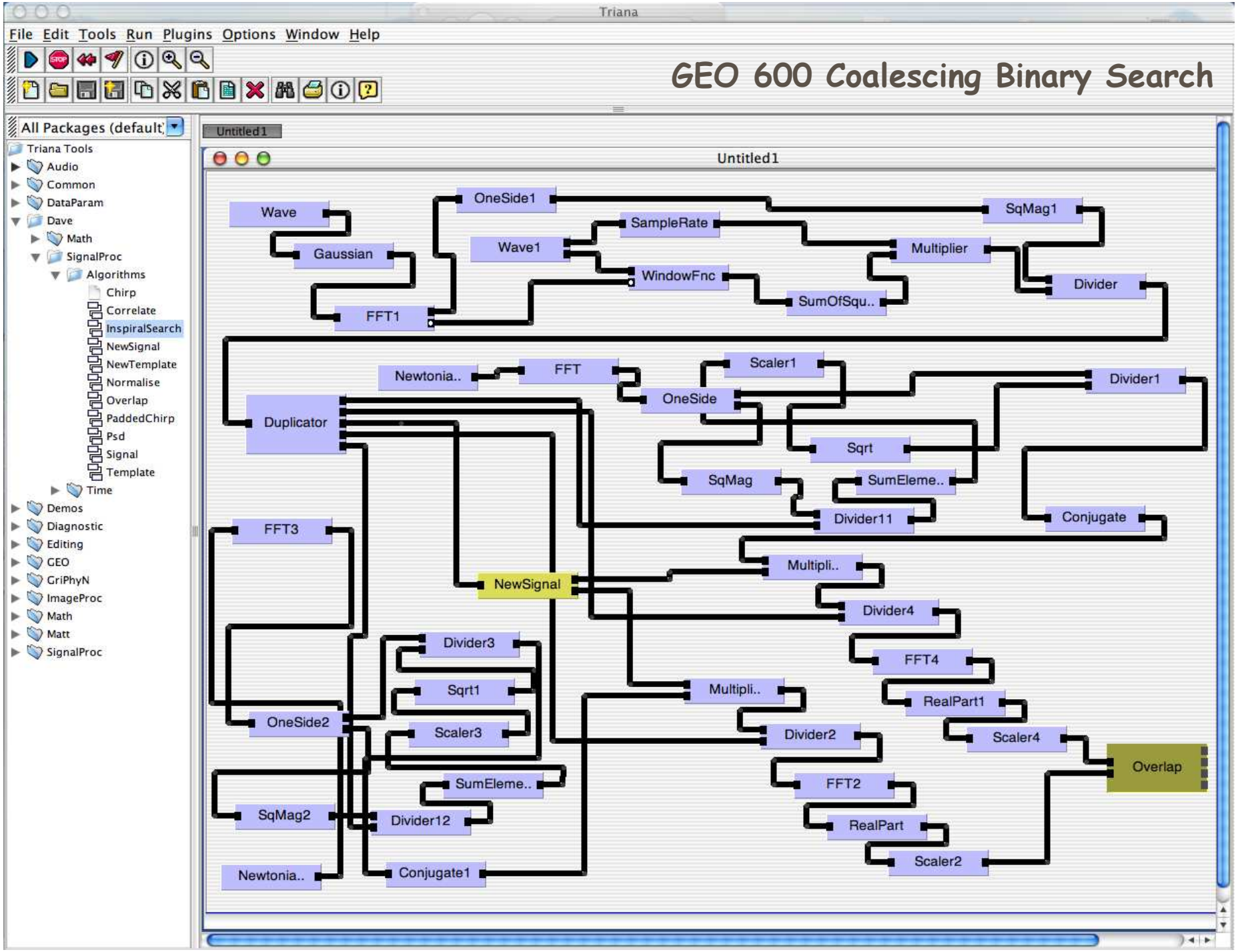
- Running services are automatically discovered via the GAP Interface, and appear in the tool tree

- User can drag remote services onto the workspace and connect cables to them like standard tools (except the cables represent actual JXTA/P2PS pipes/WSIF SOAP invocations)

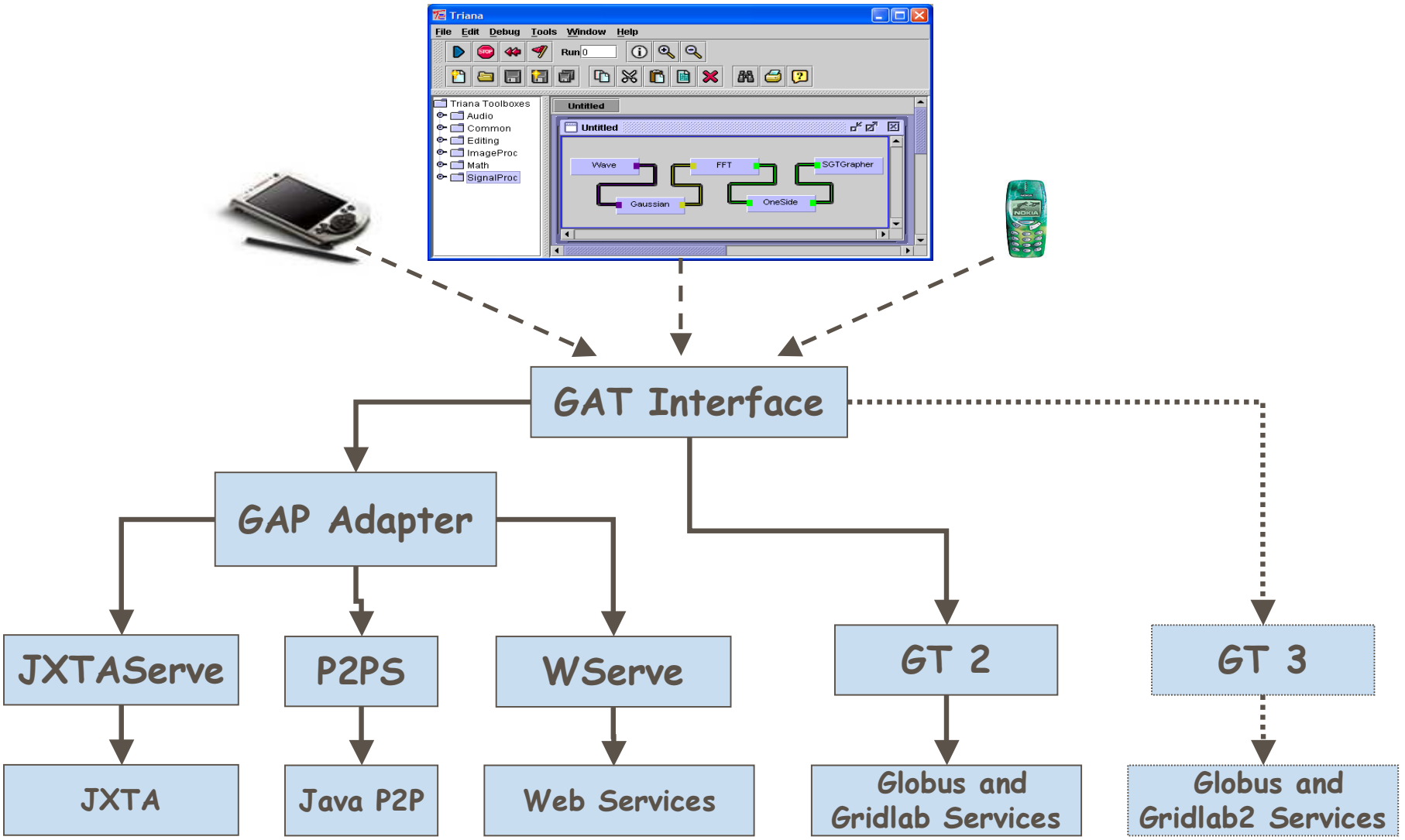


Coalescing Binary Search




Triana
File Edit Tools Run Plugins Options Window Help
GEO 600 Coalescing Binary Search
All Packages (default)
Triana Tools
Audio
Common
DataParam
Dave
Math
SignalProc
Algorithms
Chirp
Correlate
InspiralSearch
NewSignal
NewTemplate
Normalise
Overlap
PaddedChirp
Psd
Signal
Template
Time
Demos
Diagnostic
Editing
GEO
GriPhyN
ImageProc
Math
Matt
SignalProc
Untitled1
Untitled1
Wave
OneSide1
SqMag1
SampleRate
Gaussian
Wave1
Multiplier
WindowFnc
Divider
SumOfSqu..
FFT1
Newtonia..
FFT
Scaler1
Divider1
OneSide
Duplicator
Sqrt
SqMag
SumEleme..
Divider11
Conjugate
FFT3
Multipli..
NewSignal
Divider4
Divider3
FFT4
Sqrt1
Multipli..
RealPart1
OneSide2
Scaler3
Divider2
Scaler4
Overlap
SumEleme..
FFT2
SqMag2
Divider12
RealPart
Scaler2
Newtonia..
Conjugate1

Current GAP Implementations



GAT Status

- GAT Engine finished C & Java
- C: adapters to GT2 – due End October
- Java: Adapters to GT2 + P2P GAP bindings by end of October
- Gridlab2:
  - GT 3 support
  - Full P2P support
  - Application integration – 5+ areas