Infosphere:
A Midterm Update on Infopipes

Calton Pu
Professor and John P. Imlay, Jr. Chair in Software
Georgia Institute of Technology

Infosphere cercs

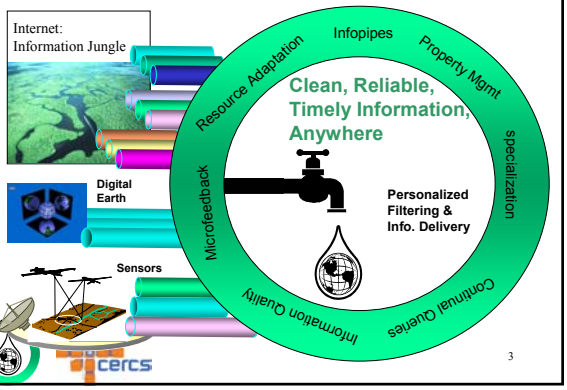
Ubiquitous Computing

- Plenty of computers (Moore's Law)
  - Are everywhere, all connected, almost free
  - Know Everything
  - 5 IT Expeditions in 1999 (e.g., MIT's Oxygen)
- Information is where value is
  - Paying distributed applications (DL, EC)
  - Cheap disks connected by cheap bandwidth

Infosphere cercs

Problem: too many sources, too much information

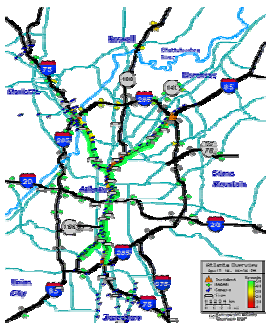
Infosphere



Infosphere cercs

NAVIGATOR

- **Freeway Surveillance**
  - 80+ PTZ Cameras
  - 500+ VDS Cams
  - Many info sources
- Potential direct input
  - Instrumented cars
  - Trucks and others



Infosphere cercs

RT Traffic Optimization

- Central traffic control scenario
  - Car navigational system registers route
  - Control sends back detailed traffic info/estimate
  - When rerouting: register the new route
- Peer-to-peer scenarios
  - Taxi/delivery trucks exchange information
  - Opposite direction cars know what's ahead

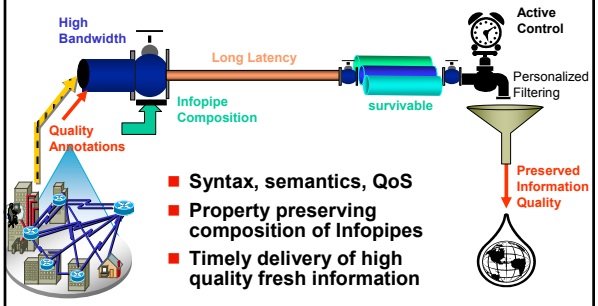
Infosphere cercs

Emergency Response

- Increasing importance after 9/11, Iraque
- Proactive planning, handling, response
  - Hurricanes, tornados, presidents, etc
  - Real-time observation and forecasting of exact route for "big events"
  - Get out of harm's way, get back to help
  - A few minutes can make a big difference

Infosphere cercs

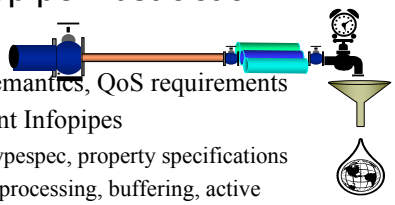
Infopipes: Backbone of Infosphere



- **Syntax, semantics, QoS**
- **Property preserving composition of Infopipes**
- **Timely delivery of high quality fresh information**

Infosphere cercs

Infopipe Abstraction



- Syntax, semantics, QoS requirements
- Component Infopipes
  - Ends: Typespec, property specifications
  - Middle: processing, buffering, active
- Composition of Infopipes
  - End-to-end QoS property preservation

Infosphere cercs

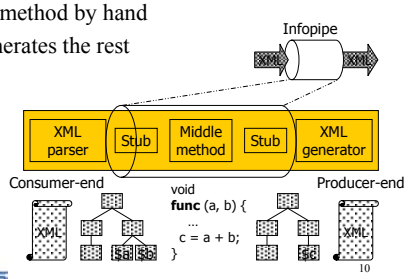
Goals for ISL/ISG

- ISL – Infopipe Specification Language
  - Simple description for Infopipes
  - Support datatypes
  - Support serial composition
  - QoS requirements
- ISG – Infopipe Stub Generator
  - Generate datatypes, communication stubs
  - Support multiple communication layers
  - Support multiple implementation languages

Infosphere cercs

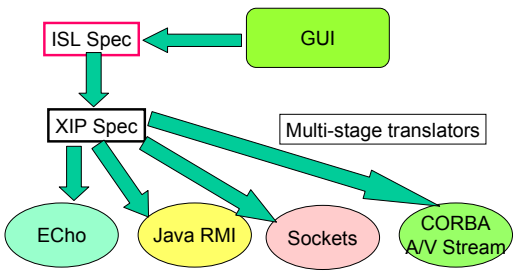
Infopipe Internals

- Information flow driven
  - Middle method by hand
  - ISG generates the rest



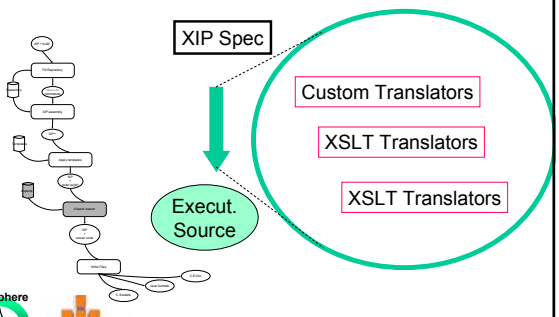
Infosphere cercs

Infopipe Software Architecture



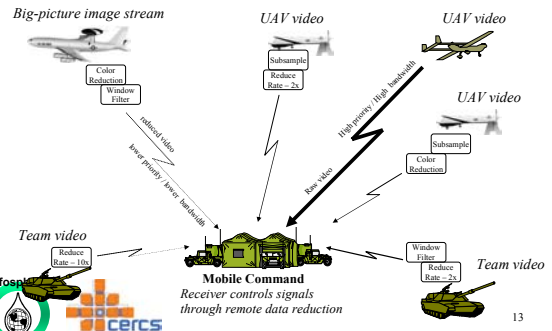
Infosphere cercs

Multi-Stage Translation

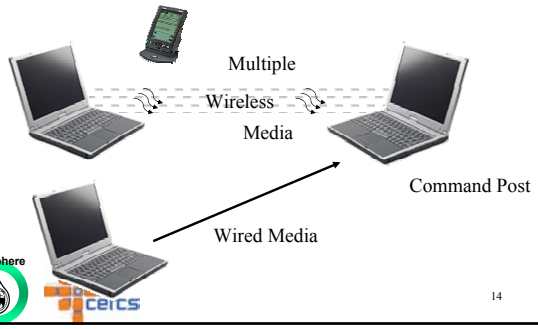


Infosphere cercs

Demo Scenario

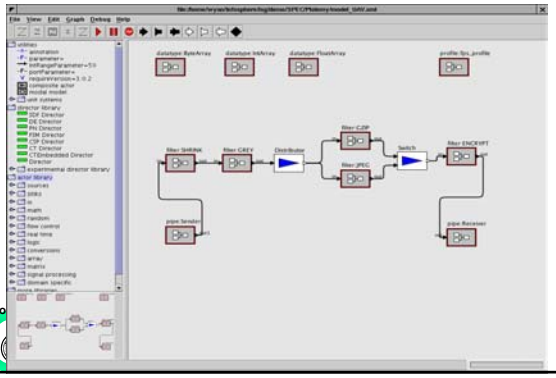


Demo Application – Sensor Streams in Wired and Wireless Environments

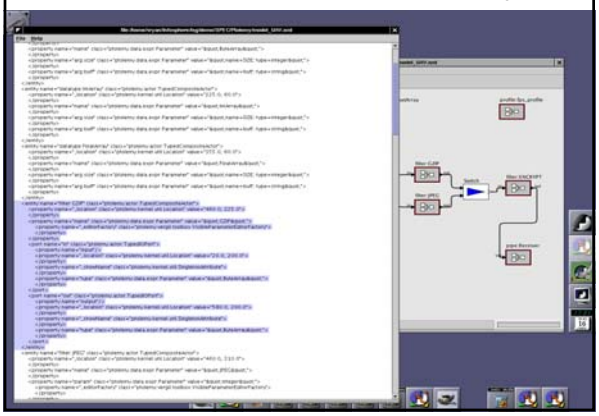


Ptolemy Design Tool

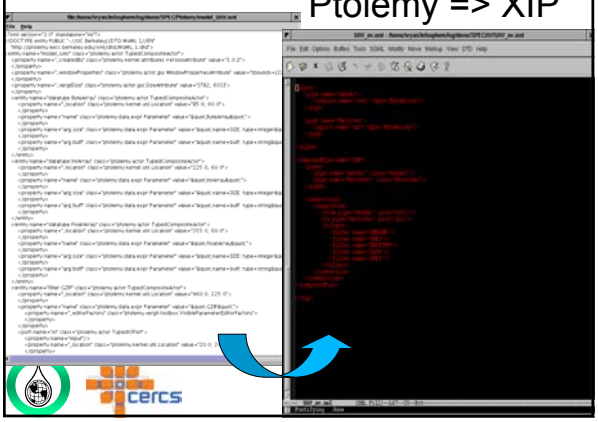
UAV: datatype, filter, pipe



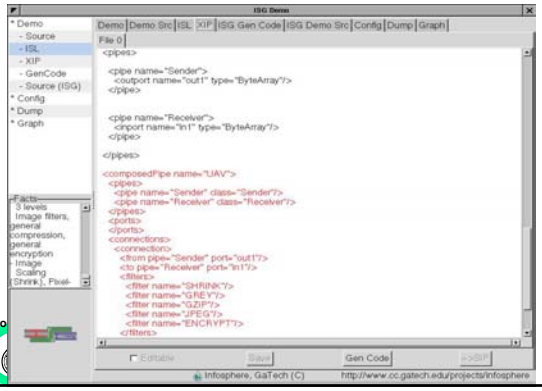
XML description



Ptolemy => XIP

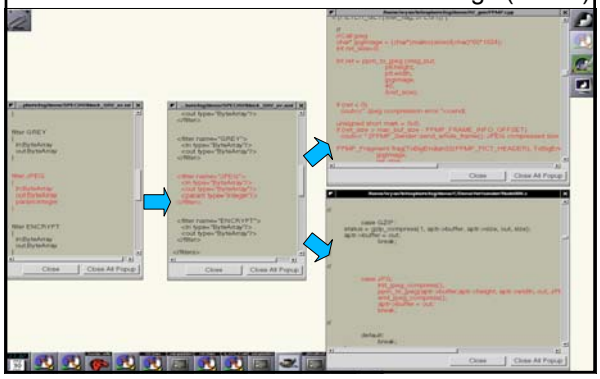


XIP => Gen code

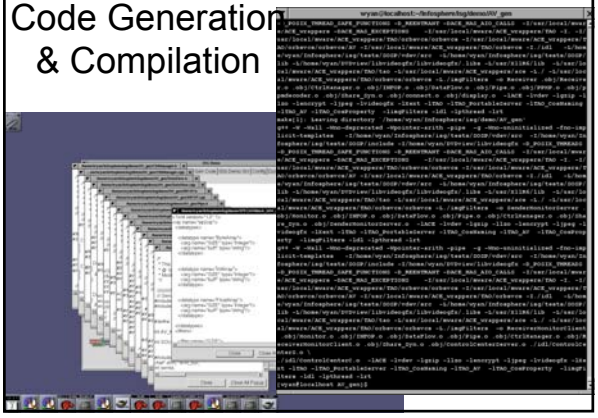


Different language bindings & different API bindings

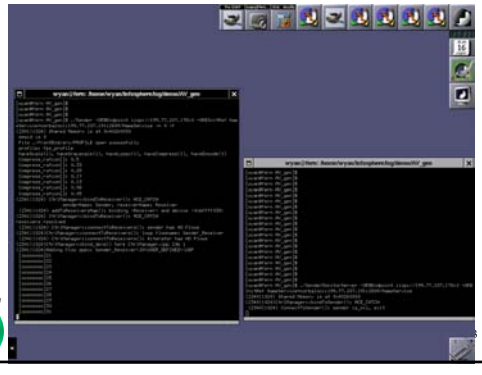
Different language bindings & different API bindings (cont'd)



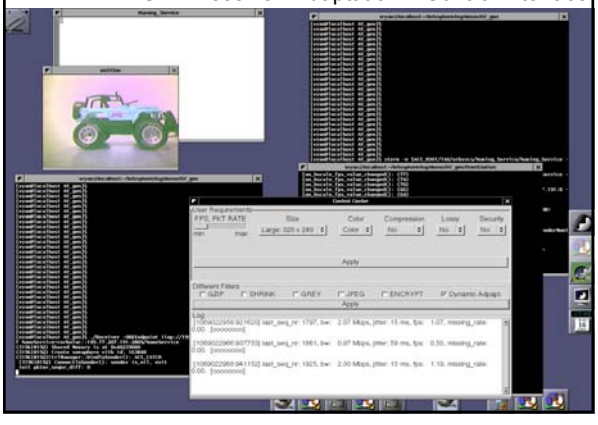
Code Generation & Compilation



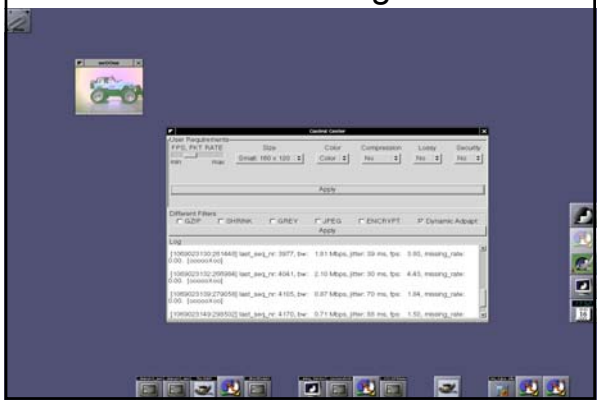
UAV Sender



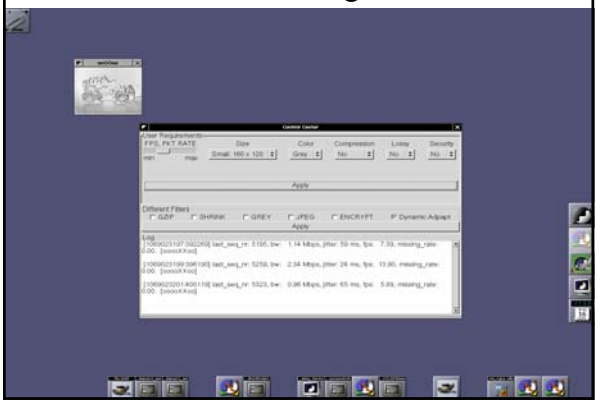
UAV Receiver: Adaptation + Control Interface



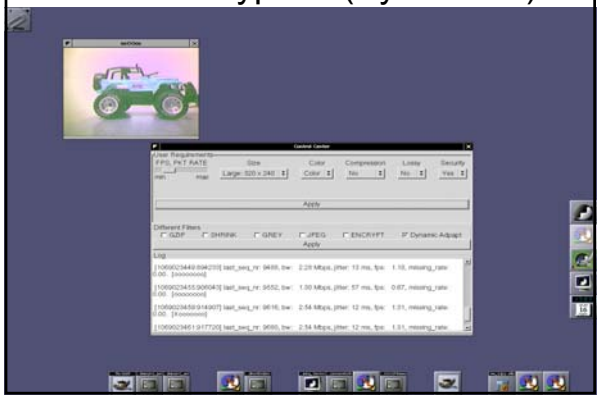
Scaling filter



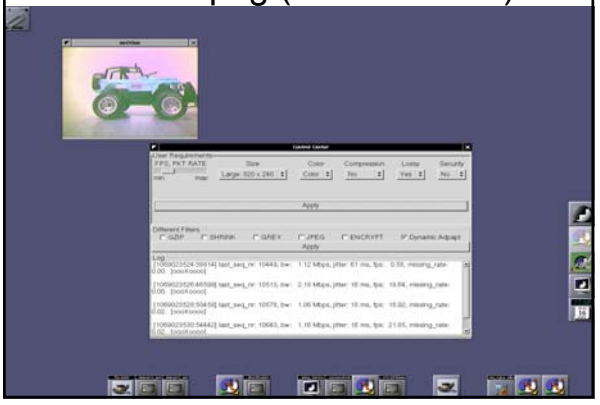
Scaling + B/W



Encryption (Symmetric)



Jpeg (factor of 1/25)



Infopipe and Software Eng.

- QoS/QoI support implemented with AOP
  - QoS dimensions: performance, security, etc
  - Performance monitoring, recording, adjusting
- Feedback-based QoS maintenance
  - Monitoring at each stage of information flow
  - Make adjustments when QoS exception raised

Infosphere cercs

XIP: An Intermediate Format

- Lingua Franca – decouples ISL from ISG
- XIP supports
  - Datatypes
  - Filters
  - Pipes
  - Serial pipe composition

Infosphere cercs

XSLT Benefits

- Modularity (xsl:include)
- Correct software
- Widely supported
- Leveraged XML extensibility
  - Allowed for abstract machines to work from common specs, but generate custom code
  - Allowed easy addition of AXpect

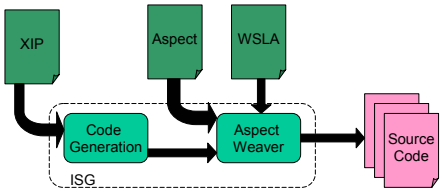
An Aspect

Pointcut: selects joinpoint in XIP from generated code

Joinpoint: Added in aspect, or from generated code

```
<xsl:template
 match="//filledTemplate[@name=$pipename][@inside=$inside]//jpt:pipe-middle">
 struct timeval base;
 struct timeval end;
 <jpt:time-process>
 // take timing here
 gettimeofday(&amp;base,NULL);
 <xsl:copy>
   <xsl:apply-templates select="@*|node()"/>
 </xsl:copy>
 gettimeofday(&amp;end,NULL);
 usec_to_process = (end.tv_sec - base.tv_sec  )*1e6+(end.tv_usec - base.tv_usec);
 fprintf(stdout,"time to process: %ld\n", usec_to_process);
 </jpt:time-process>
</xsl:template>
```


AXpect Weaver



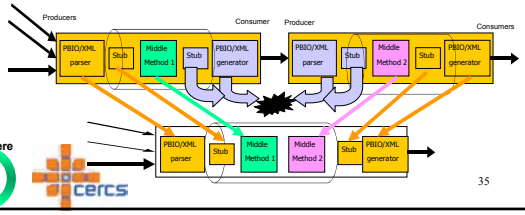
- XIP specification has `apply-aspect` commands
- Aspects are XSLT, operate on XIP + generated code
- Aspects can refer to outside XML (e.g. WSLA) for additional data
- Source code is C, C++, Java

An AOP Experiment

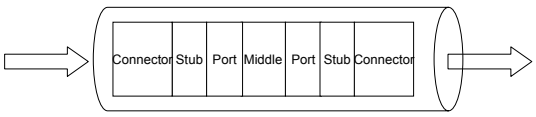
- Infopipe: Image stream sender to receiver
- Receiver has CPU usage limits
- Aspects build on top of other aspects (e.g. CPU usage computation requires timing info)
- Rate control aspect governs receiver CPU usage through sender rate adjustment
- Aspect code accounts for 348 of 1135 NCLOC (31%)
- Aspects affect 9 of 14 generated files

Specialization of Infopipes

- Specialization of Infopipes
  - Tools: Tempo-C and Java partial evaluator
  - Goal: reduce interpretation of data (2nd example later)

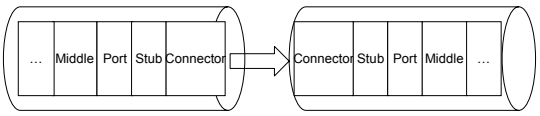


Infopipe Further Details



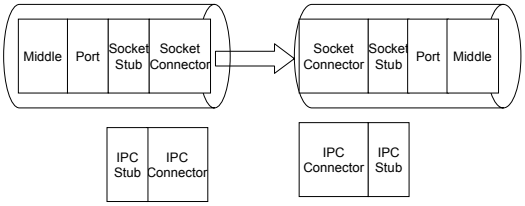
- Middle : Pipe's processing module
- Port : Communication endpoint abstraction
- Stub : Data marshalling
- Connector : Communication runtime, Handles connection info

Pipe communication

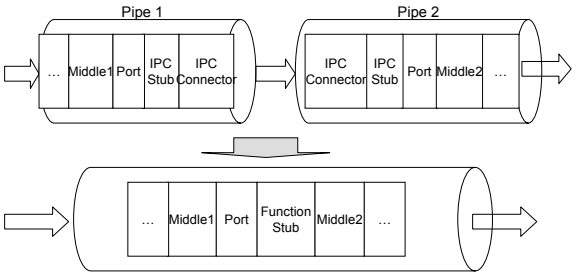


- Each port can be plugged with different types of stubs and connectors
- Currently we have 3 types
  - Socket connector
  - IPC(unix pipe) connector
  - Function connector

Replugging (1)



Replugging (2)



- Pipe 1, 2 are collapsed into one pipe
- Pipe 1 loads Pipe 2's middle code dynamically

Benchmark results

- Send 8 byte data (in the same machine)

Socket	41.7 us
IPC	12.6 us
Function	1.8 us

- Specialization cost

Socket->IPC
1127.6 us
IPC->Function
1045.7 us

Code examples (pseudo code)

- Port has the same interface

```
pipeSource->outPorts["out1"]->push(item);
```

```
Port::push(item)
{
        lock(stub);  // for replugging synchronization
        stub->push(item);
        unlock(stub);
}
```

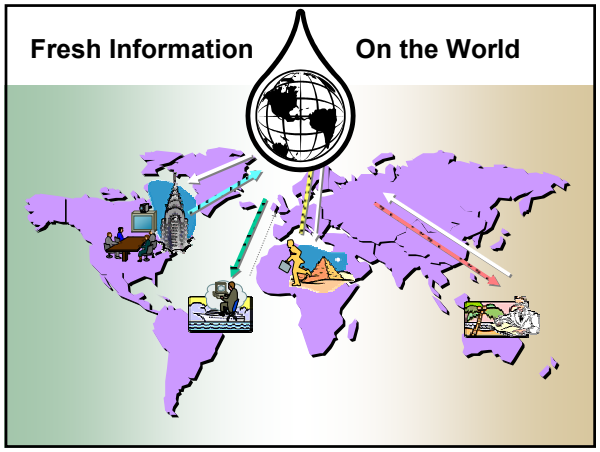
```
StubSocket::push(item)
{
    marshal(item, buf);
    conn->send(buf);
}
ConnSocket::send(buf)
{
    send(sock, buf);
}
```

```
StubIPC::push(item)
{
    marshal(item, buf);
    conn->send(buf);
}
ConnIPC::send(arg)
{
    write(pipe_fd, arg);
}
```

```
StubFunction::push(item)
{
    nextPipe->middle
            ->process(item);
}
```


Project Summary

- Infopipe: Distributed information flows
  - ISL, GUI, XIP
- ISG and software engineering tools
  - AOP and AXpect
  - Specialization of Infopipes
- End-to-end QoS properties
  - Performance, maintainability, scalability


Fresh Information
On the World