

I A M C

Paul S. Wang
Institute for Computational Mathematics
Kent State University

`pwang@mcs.kent.edu`

`http://monkey.mcs.kent.edu/~pwang`

Contents

- Goals and Motivations
- Overview of IAMC Architecture
- MCP (Mathematical Computation Protocol)
- IAMC client – a MCP client
- IAMC server – a MCP server
- IAMC client-server communication
- IAMC Server to compute engine interface

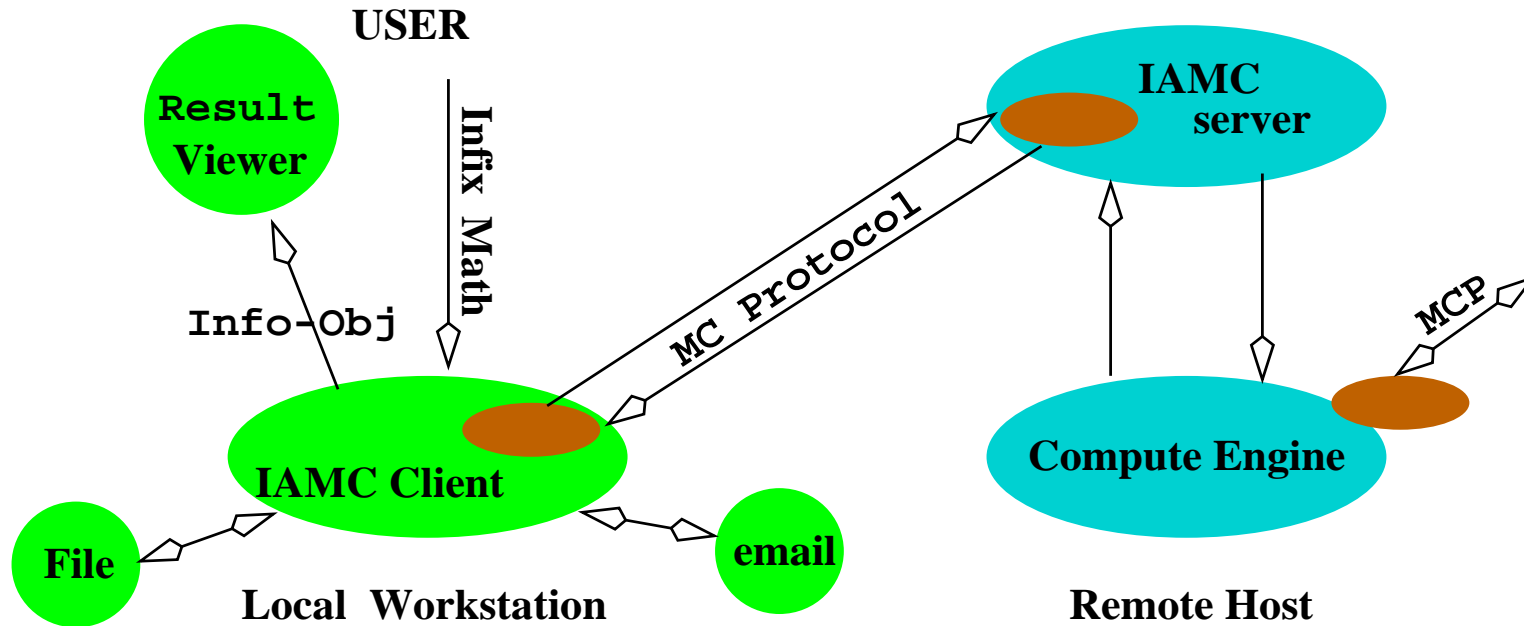
Mathematics on the Web

- Table of integrals (socket connection, UC Berkeley)
- Demos on SymbolicNet (CGI, X, Kent State)
- MathML (W3 Consortium)
- NetSolve (U. Tennessee and Oak Ridge National Lab)
- Techexplorer (IBM)
- WebEQ (Geometry Technologies Inc)

Goals and Motivations

- Many kinds of info and services are easily accessible on the Internet
- Sharing of mathematical/scientific results is lagging behind
- To make math-oriented data and services easily accessible on the Internet – directly, via Web, and by email.
- To allow remote *compute servers* usable almost like local programs
- To allow effective and efficient communication of mathematical data over the Internet
- To make it possible to heterogeneous compute servers to exchange computational results and perform further computation on them.
- To make IAMC compatible with prevailing technologies
- To make IAMC convenient, simple to use, and easy to learn.

IAMC Architecture Overview



IAMC Applications

- Use of remote computation services
- Access remote scientific databases
- Making parallel/super computing accessible
- Distance learning
- Computing via NetPC for high school or occasional users.
- Problem solving environments (PSE)

Leveraging Prevailing Technologies

- Internet and the Web
- MathML – as part of HTML
- MP – a binary math data encoding/transfer protocol
- OpenMath – evolving math representation and semantics ‘standard’ largely by the European Maple group.
- Java – platform independence and network/GUI able
- Java RMI and/or Corba – for distributed applications

IAMC Service Overview

- Web-based — IAMC service providers will publish and make their services available with Web documents

- IAMC addresses — links to IAMC servers in the form

`iamc://hostname:port/service-id`

`http://hostname/.../cgi-bin/amac-gateway`

- IAMC clients are clients that can be launched independently or by Web browsers.

- An IAMC server will perform the service
- Connection between IAMC client can be TCP/IP, persistent HTTP, regular HTTP/CGI, or email based.
- IAMC client may perform simple one-transaction computations, longer connection-based computations, connect to multiple servers at the same time.
- IAMC server may start and control a separate compute engine.

Email-based IAMC

- Request format:

Subject: IAMC REQUEST *version*
Content-Type: application/x-mp
Transfer-Encoding: base64
To: *serverid.IAMC@host*

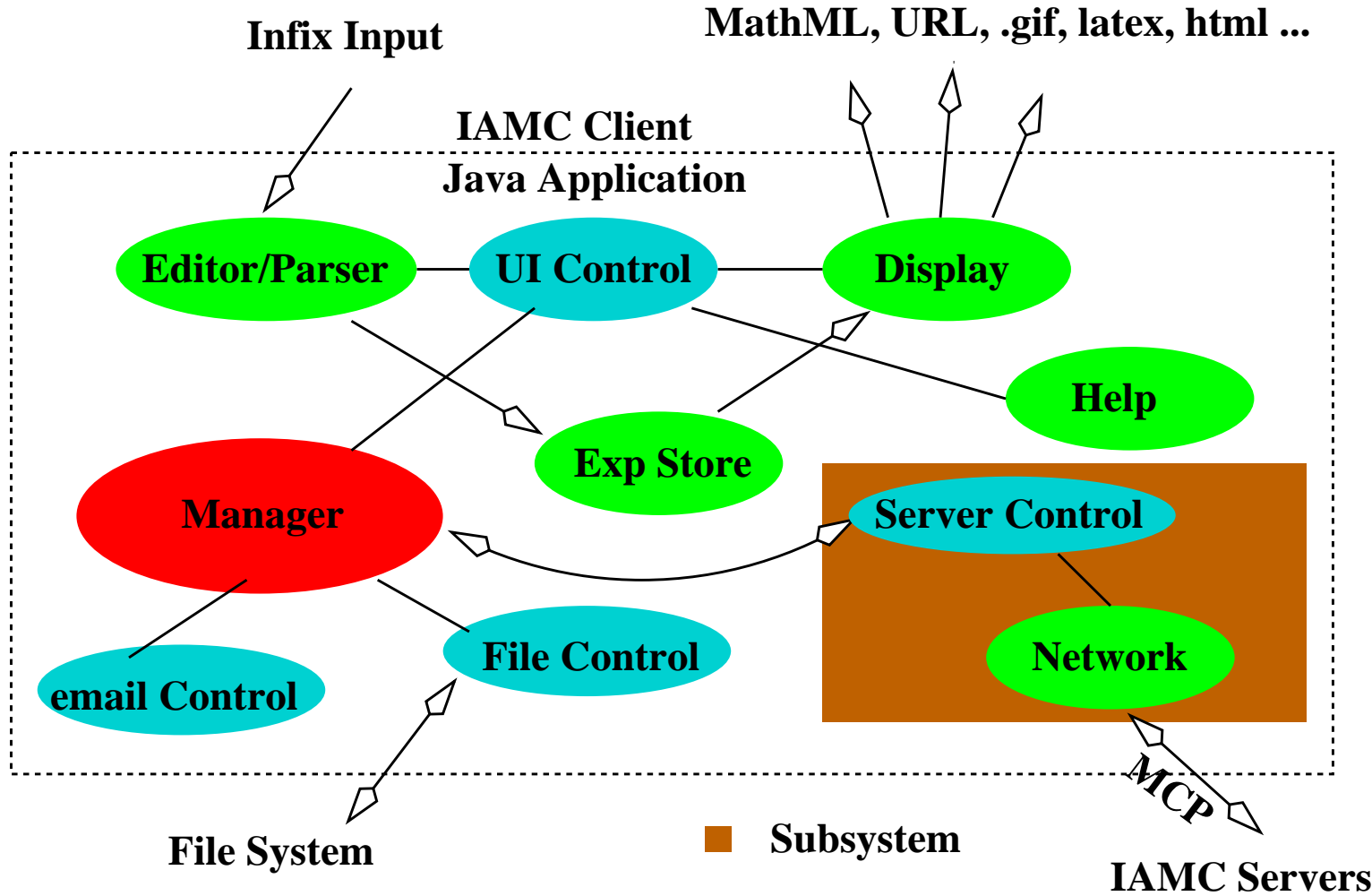
- Response format:

Subject: IAMC RESPONSE *status*
Content-Type: application/x-mp
Transfer-Encoding: base64
To: *reply address*

CGI-based IAMC

- Use HTTP POST query
- Use Content-Type: application/x-mp
- Send body of query/response without encoding

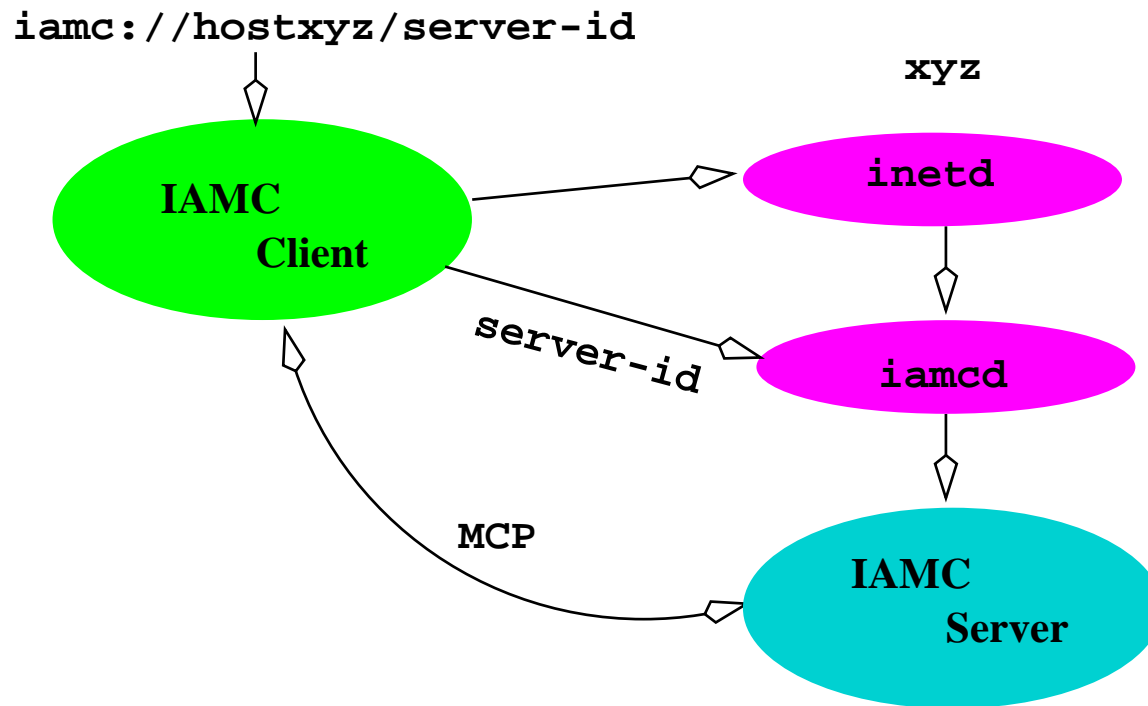
IAMC Client Design



IAMC Client Responsibilities

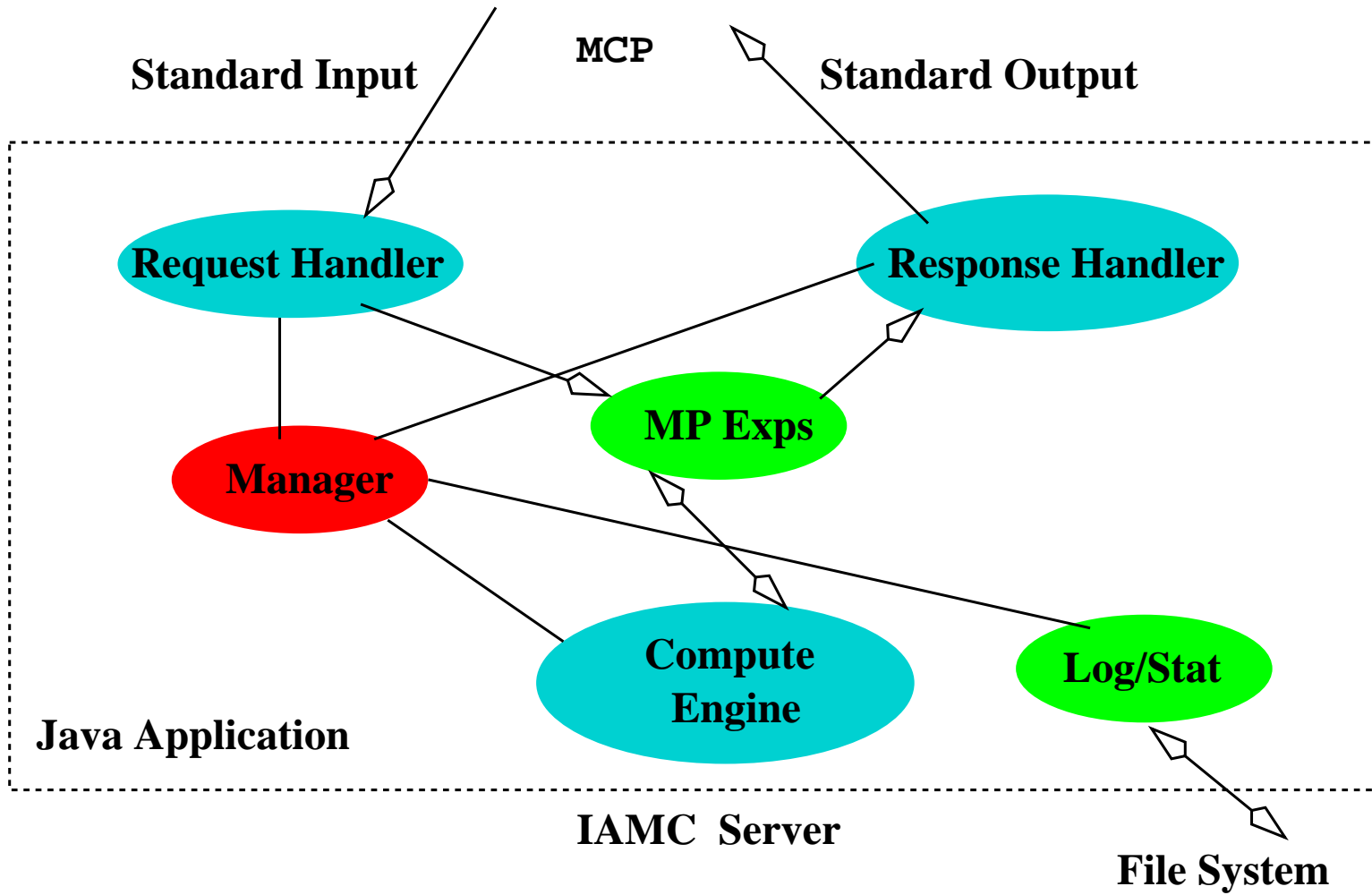
- Receives input from user (infix, menu, button, ...), and/or file
- Produces display-able results: MathML, points, .gif, URL, HTML, infix, prefix, latex, ...
- Controls one-time transaction or user session
- Provides server-supplied help/documentation info to user
- Reads/edits infix math
- Keeps track of user input and history
- Stores-retrieves files from local file system
- Connects to one or more remote servers
- Communicates with remote server in MCP
- Provides email interface

Connecting to an IAMC Server



IAMC Server Design

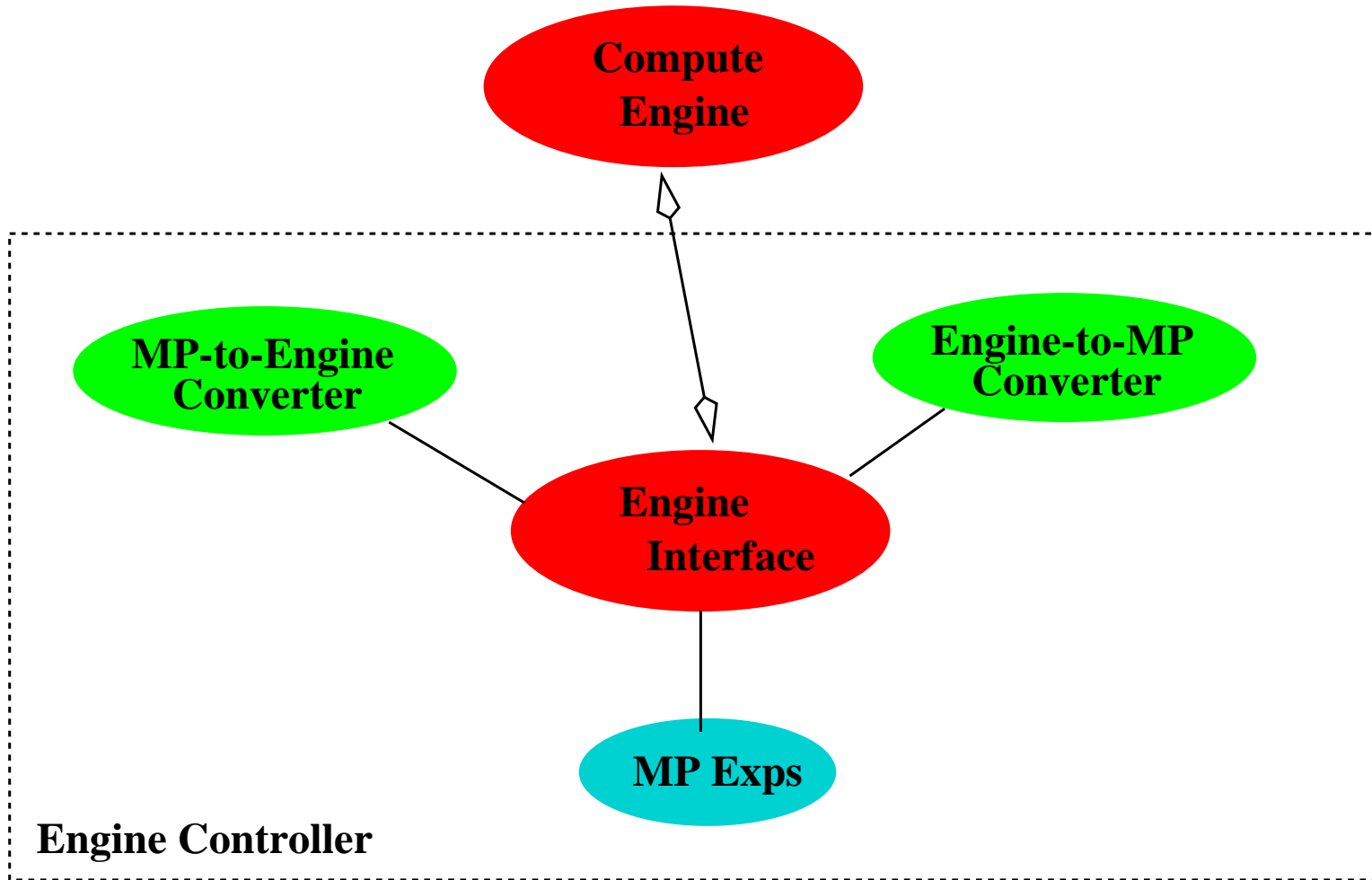
IAMC Client or Agent



IAMC Server Responsibilities

- Receives IAMC request from client from `stdin`
- Sends IAMC response to client to `stdout`
- Performs requested operation or computation
- Maintains state
- Provides user help and usage documentation through client
- Keeps log of operations, errors, and statistics
- Speaks MC protocol
- Spawned by `iamcd`, `cgi`, or email processing.

Interface to External Engine



MCP: Mathematical Computation Protocol

- Addressing and making connections
- Single-transaction vs. stream connection
- Compatible with `tcp`, `http` and email
- MCP request and response formats
- MP encoding of mathematical data
- Computation, control, help, requests
- Server-side *prompts*
- Synchronous vs. asynchronous modes
- Session-control functions such as interrupt, reset

MP: A Mathematical Data Encoding/Transfer Format

- Mathematical data represented as *binary-encoded* parse trees
- Annotations
- Dictionaries
- Transport layer independence
- Efficiency and flexibility
- C language support (MP-1.1.3)

MP Binary Encoding Example

$(f := x \rightarrow x * *3 - 1)$ _(source maple)

Type	Dict	Value	Annot	Args
Op	ElemAlg	FncAssign	1	3
Annot	MP	Source		
String	Session	Maple	0	
Op	ElemAlg	FncName	0	1
Id	Session	f	0	
Op	ElemAlg	FormalArgs	0	1
Id	Session	x	0	
Op	ElemAlg	FncDefinition	0	1
Op	ElemAlg	Subtract	0	2
Op	ElemAlg	Power	0	2
Id	Session	x	0	
Int	--	3	0	
Int	--	1	0	

MathML Presentation Markup

The expression $x^2 + 4x + 4 = 0$ is coded as follows

```
<mrow>
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
  <mo>+</mo>
  <mn>4</mn>
  <mi>x</mi>
  <mo>+</mo>
  <mn>4</mn>
  <mo>=</mo>
  <mn>0</mn>
</mrow>
```

Another Presentation Example

$$\sqrt[3]{1 - \frac{x}{2}}$$

```
<mroot>  
  <mrow>  
    <mn>1</mn>  
    <mo>-</mo>  
    <mfrac>  
      <mi>x</mi>  
      <mn>2</mn>  
    </mfrac>  
  </mrow>  
  <mn>3</mn>  
</mroot>
```

MathML Content Markup

MathML also employs a *prefix notation* to encode the content of the mathematical expressions:

The general form is

`<apply> Operator operand1 ... </apply>`

For example:

$\sin(x) + 9$

Markup:

```
<apply><plus/>  
  <apply><sin/><ci>x</ci></apply>  
  <cn>9</cn>  
</apply>
```

Another Content Example

$$\frac{d^3}{dx^3} f(x)$$

Markup:

```
<apply><diff/>  
  <apply><fn> f </fn>  
    <ci> x </ci>  
  </apply>  
  <bvar>   <ci>x</ci> </bvar>  
  <degree> <cn>3</cn> </degree>  
</apply>
```

$$(1, 2) \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = (2, 1)^t$$

Markup:

```
<e> <eq/>
  <apply><times/>
    <vector> <cn>1 </cn> <cn>2 </cn>
    </vector>

    <matrix>
      <matrixrow> <cn>0 </cn> <cn>1 </cn> </matrixrow>
      <matrixrow> <cn>1 </cn> <cn>0 </cn> </matrixrow>
    </matrix>
  </apply>

  <apply> <transpose/>
    <vector> <cn>2 </cn> <cn>1 </cn>
    </vector>
  </apply>
</e>
```

IAMC Client Converts MP to MathML

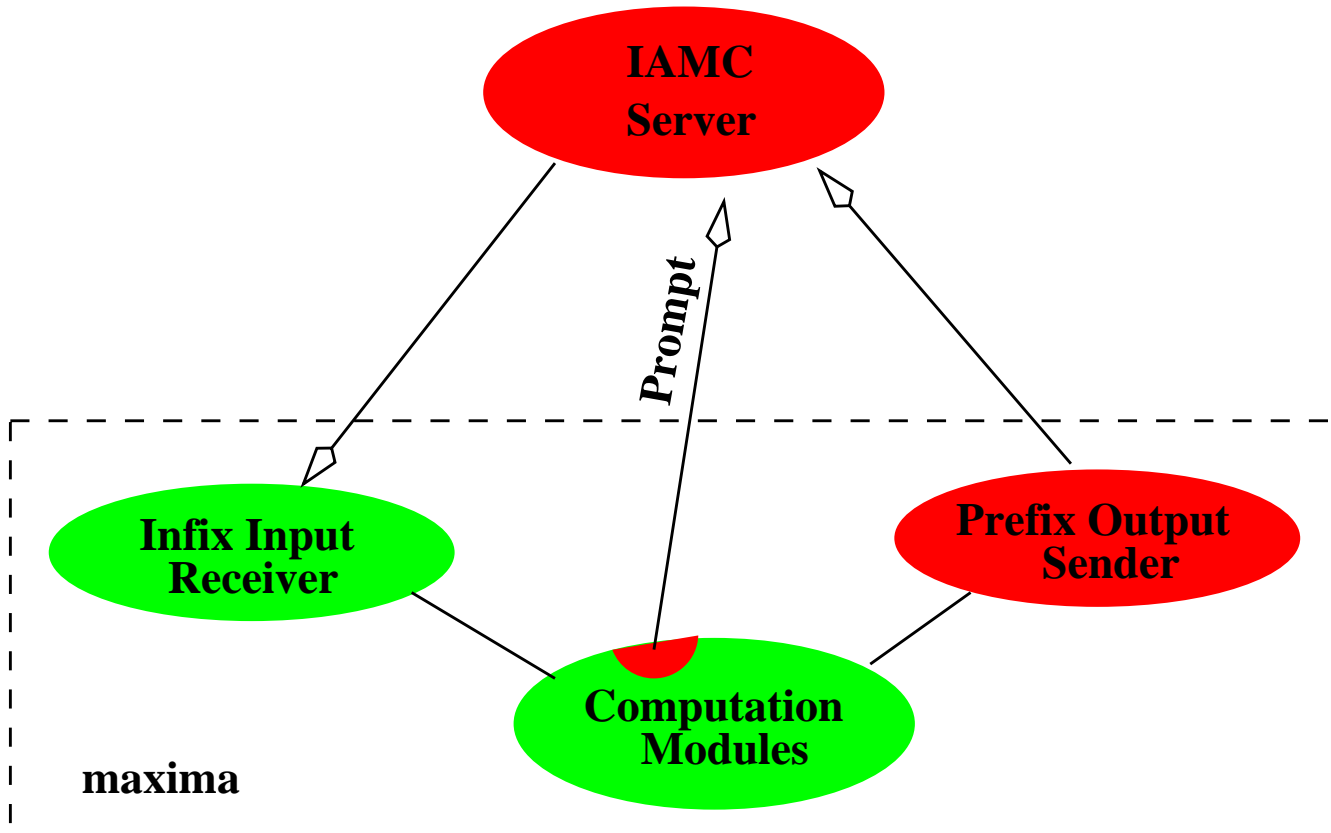
- Converting prefix MP to prefix MathML is not hard.
- Output MathML has advantages
- Leveraging over Web browser plug-ins or stand-alone MathML viewers to make nice presentations.

External Engine Interface Issues

- Assume same-host interprocess communication
- Input to engine is infix – need MP-infix converter
- Input to engine is prefix – need MP-prefix converter
- Output from engine is prefix – need prefix–MP converter
- Output from engine is infix – need infix–MP converter
- Output from engine is known content type – pass to client unchanged

- Prefix I/O to engine can be more efficient
- Infix input and prefix output can be easier to arrange
- It is possible to require *no modification* of the engine if the output is not to be converted to MP format
- Handling of dialog and error state I/O from engine needs further consideration
- Sending interrupts to engine may not be easy in many cases.

Server-MAXIMA Interface



■ Existing Code

■ New Code

Interface Scenario

1. IAMC server converts incoming request to infix MAXIMA input.
2. IAMC server sends input to MAXIMA.
3. MAXIMA performs the computation.
4. MAXIMA outputs answer in prefix form to IAMC server.
5. IAMC server converts answer to MP.
6. MAXIMA sends ready prompt
7. IAMC server sends next request to MAXIMA

Modification to MAXIMA

```
(defun displa(exp)
  (print 'BEGIN-EXP)
  (print exp)
  (print 'END-EXP)
  (terpri)
)
```

Effect of the Modification

```
(C1) loadfile("newdispla.lsp");
```

```
Loading newdispla.lsp
```

```
Finished loading newdispla.lsp
```

```
BEGIN-EXP
```

```
((MLABLE) $D1 $DONE)
```

```
END-EXP
```

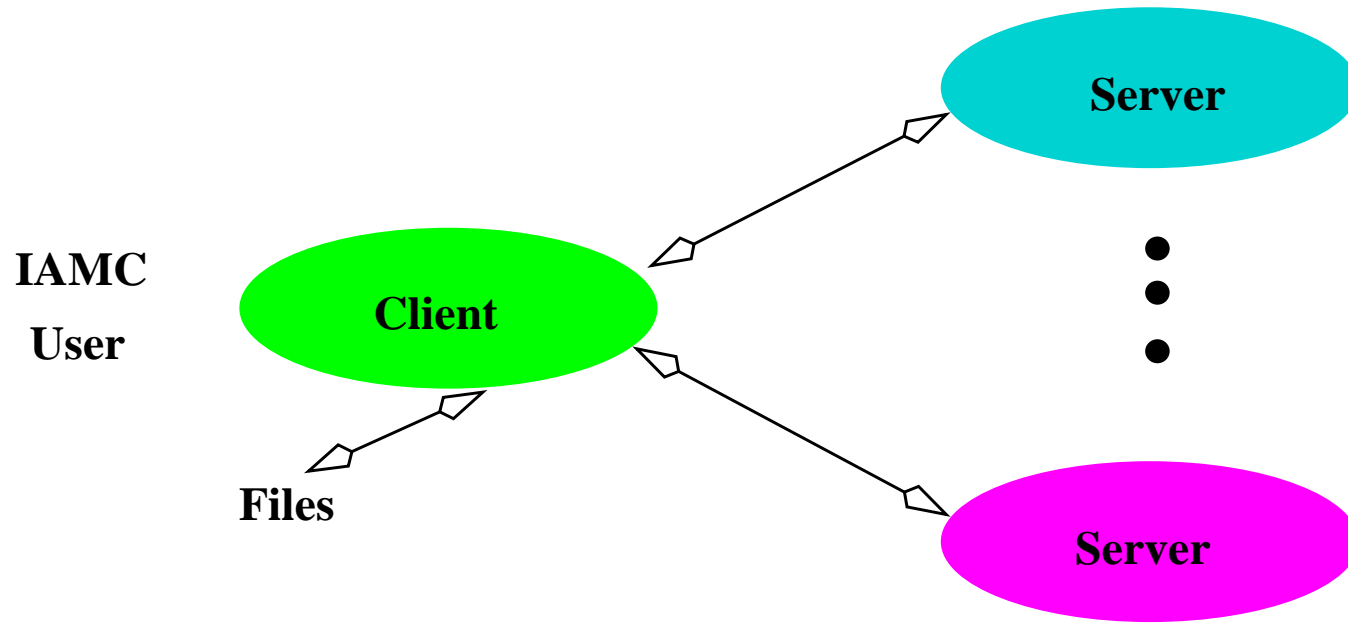
```
(C2) diff(sin(x),x);
```

```
BEGIN-EXP
```

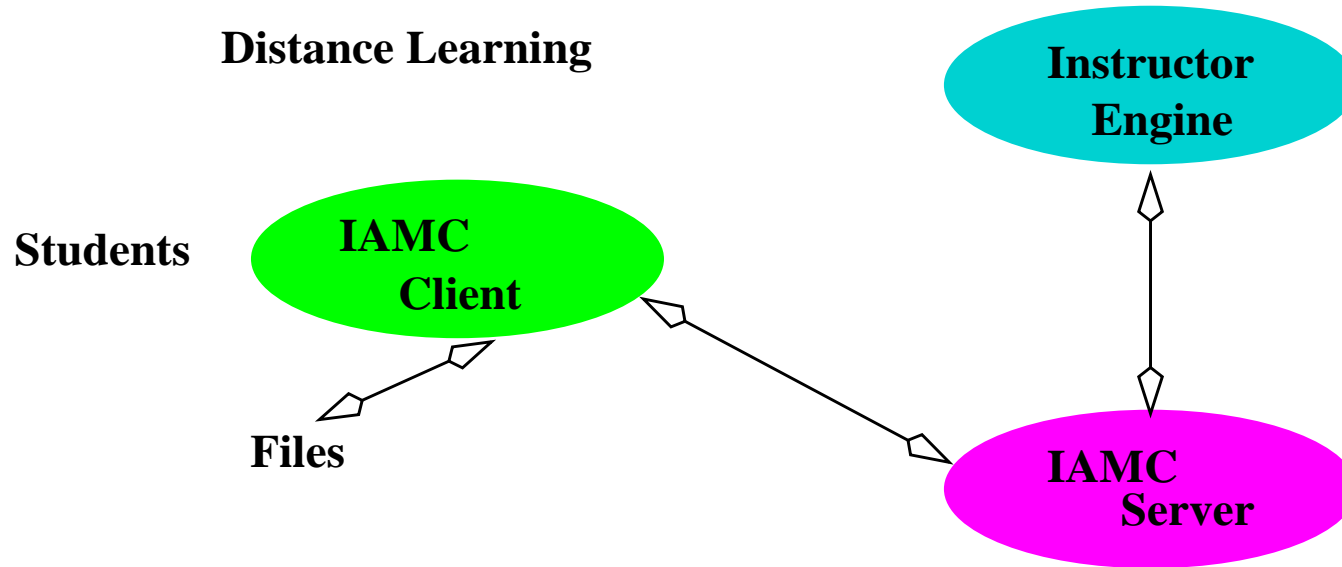
```
((MLABLE) $D2 ((%COS SIMP) $X))
```

```
END-EXP
```

```
(C3)
```



Distance Learning



Problem Solving Environments

