FIPA-OS

Open Source
FIPA Agent Platform

by Milla Mäkeläinen

Overview

- Open source benefits
- FIPA reference model
- Example application: FACTS project
- Conversation management
- FIPA-OS class structure
- Example agent

FIPA: Situation now

- FIPA specifications are available, but - until recently - no **reference implementation**
- Validation and verification of FIPA restricted
- FIPA feedback and maintenance an issue
- FIPA still not adopted widely

Open Source Benefits

- Collaborative and co-operative approach
- FIPA Open Source provides:
  - Open source model for FIPA
  - Baseline implementation publicly available
  - Enable agent application developers to construct apps using FIPA technology
  - Encourage extensions/feedback/iterative/ evolving implementation(s)

FIPA Open Source

- Wider developer adoption of FIPA
- Realises the FIPA promise of interoperability - the agent paradigm
- Helps FIPA to concentrate on agent issues
- The hurdle to adopt FIPA is reduced
- Enables users to concentrate on agent-enabled business thrusts, rather than underlying platform/middleware issues

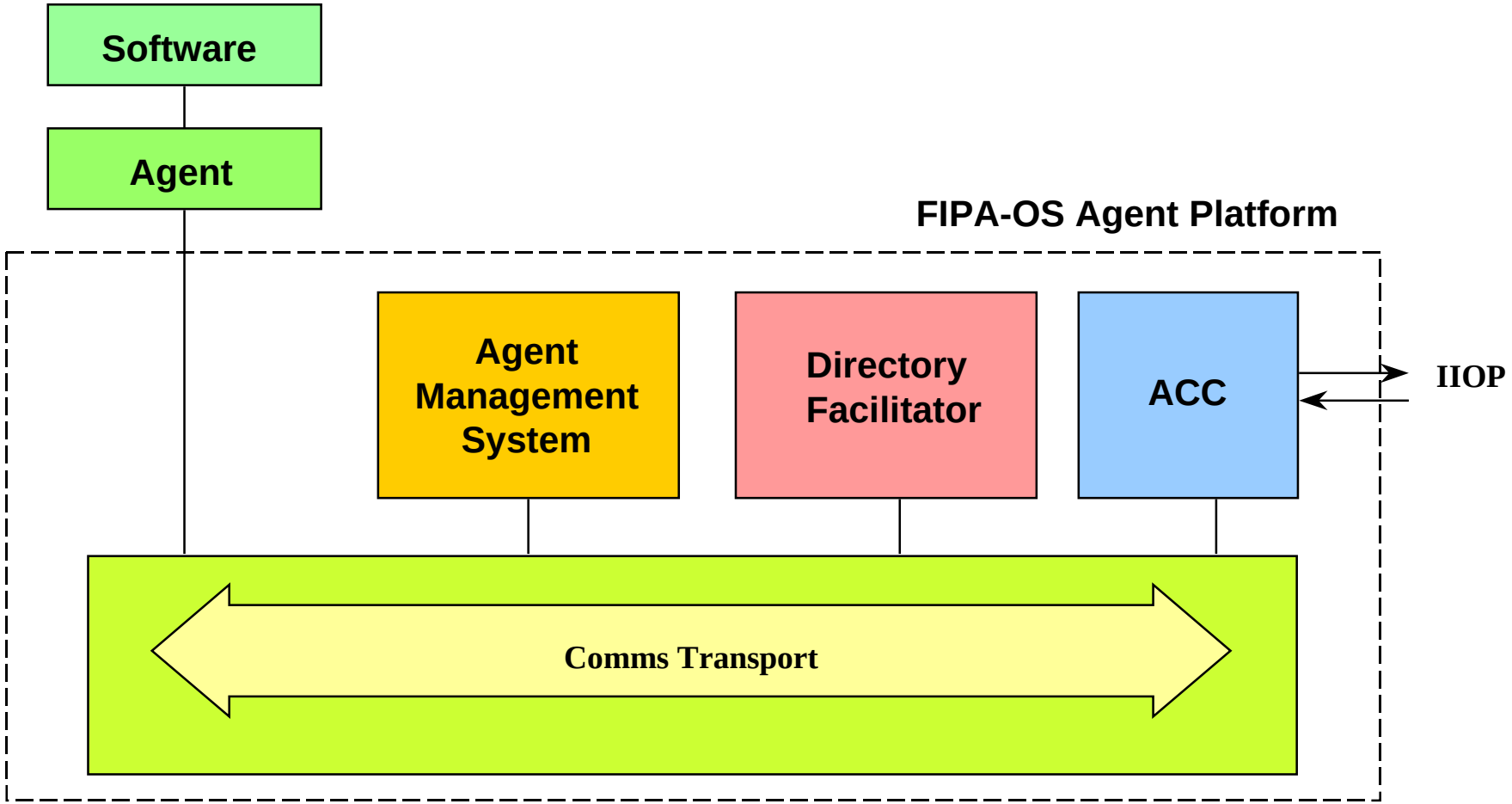
FIPA-OS

- Not owned by anyone - it is public Open Source
- Originates from Nortel Networks
  - currently manages the releases of FIPA-OS
  - this in no way precludes individuals development or exploitation plans
- FIPA-Net - FIPA DFs on the Internet
  - Imperial College have this almost ready

Other FIPA platforms

- JADE
  - Java Agent DEvelopment Framework
  - by CSELT
  - open source
  - supports FIPA97
  - sharon.cselt.it/projects/jade
- ZEUS
  - by British Telecom
  - not strictly a FIPA platform
  - supports FIPA ACL
  - www.labs.bt.com/projects/agents/research/collaborative.htm

FIPA Reference Model



FIPA-OS contents

- Platform Agents
  - AMS, DF, ACC
- Agent Shell
  - ACL, SL0 and XML\RDF parsers
  - Persistence abstract interface (bindings for serialization)
  - Transport abstract interface (bindings to Voyager and SunIDL)

FIPA-OS contains

- Configuration
  - XML\RDF Platform and Agent profiles
  - IOR distribution via HTTP

FIPA-OS requirements

- Java 2
  - JDK1.2.2
  - www.java.sun.com
- IBM XML4J
  - XML parser, version 1.1.16
  - www.ibm.com
- SiRPAC
  - RDF parser, version 1.14
  - www.w3c.org
- Optionally:
  - web server (more than one platform)
  - ObjectSpace Voyager - alternative transport

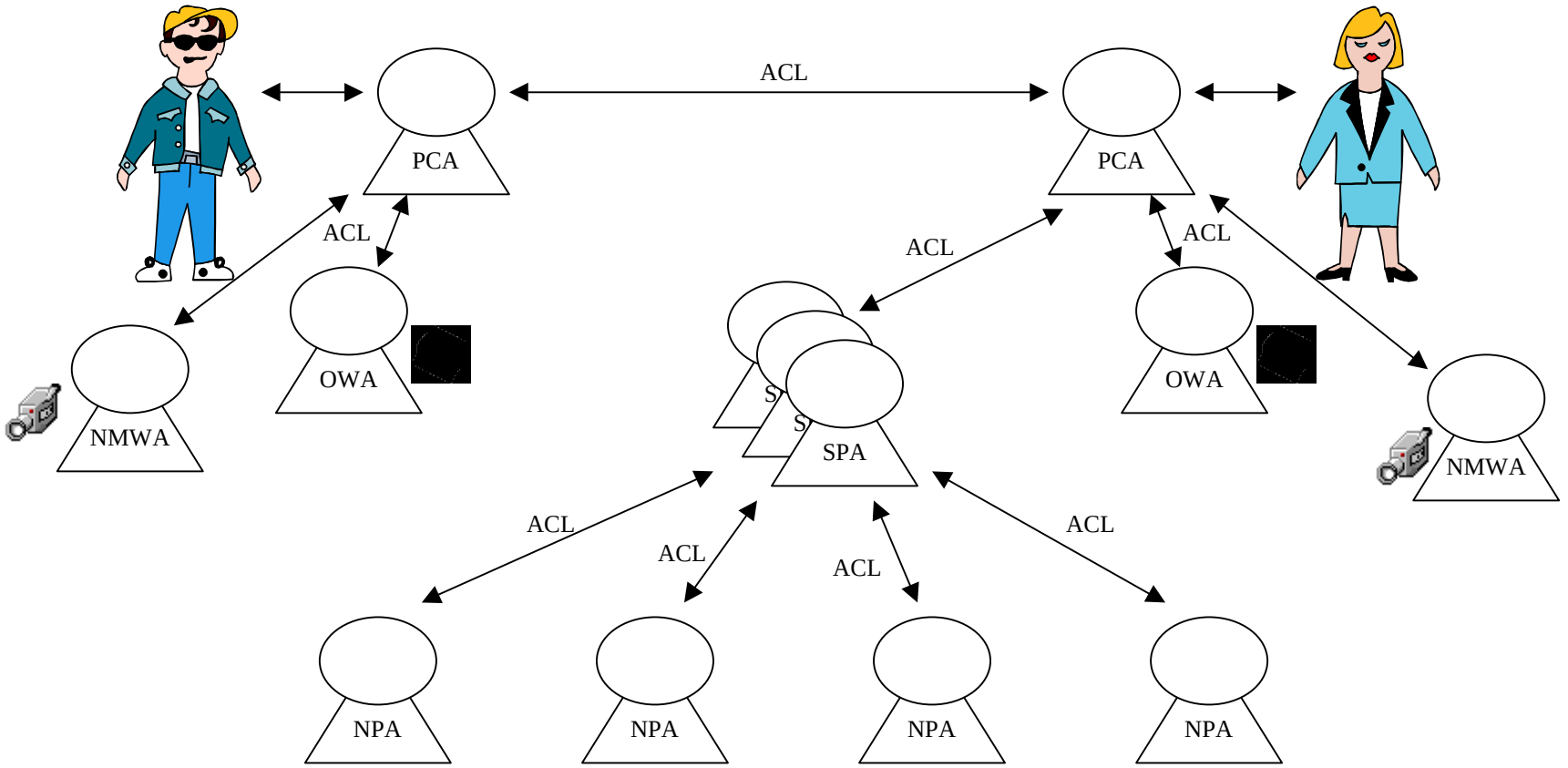
Example application: FACTS

- FACTS = FIPA Agent Communication Technologies and Services
- Virtual Private Network (VPN) scenario
  - Problem: book a video conference meeting
  - Answer: give task to your Personal Communication Agent, and stop worrying!

VPN scenario agents

- Agents:
  - FIPA platform agents + Timer Agent
  - Wrapper Agents: Microsoft Outlook Wrapper Agent (OWA) and Microsoft NetMeeting Wrapper Agent (NMWA)
  - Personal Communication Agent (PCA)
  - Service Provider Agent (SPA)
  - Network Provider Agent (NPA)

VPN scenario



User preferences

- PCA knows users preferences, so it can offer most desirable service to him/her
- User preference profile includes following information (example):
  - frame rate (high/medium/slow)
  - meeting time (morning/afternoon/not lunchtime)
  - maximum cost of meeting

PCA - SPA negotiation

- PCA negotiates with Service Providers SPA to obtain a competitive bid for the required service
- Issues include
  - frame rate
  - frame size
  - price
  - penalties for failing the service

SPA - NPA negotiation

- Service Providers (SPAs) negotiate with Network Providers (NPAs) to obtain a competitive bid for the network resources
- Issues include:
  - price
  - penalty for failing the service
  - bandwidth
  - availability

Commissioning the meeting

- Conference is automatically commissioned by the contracted Service and Network Providers at the negotiated time and place
- Client software is automatically configured by the PCA to utilise the available network resources

Extending the contract

- It’s possible to extend the contract while the meeting is on:
  - extending the meeting time
  - adding new participants
  - time and/or iterations must be limited - end users are waiting for the change to occur
  - some issues are not negotiable

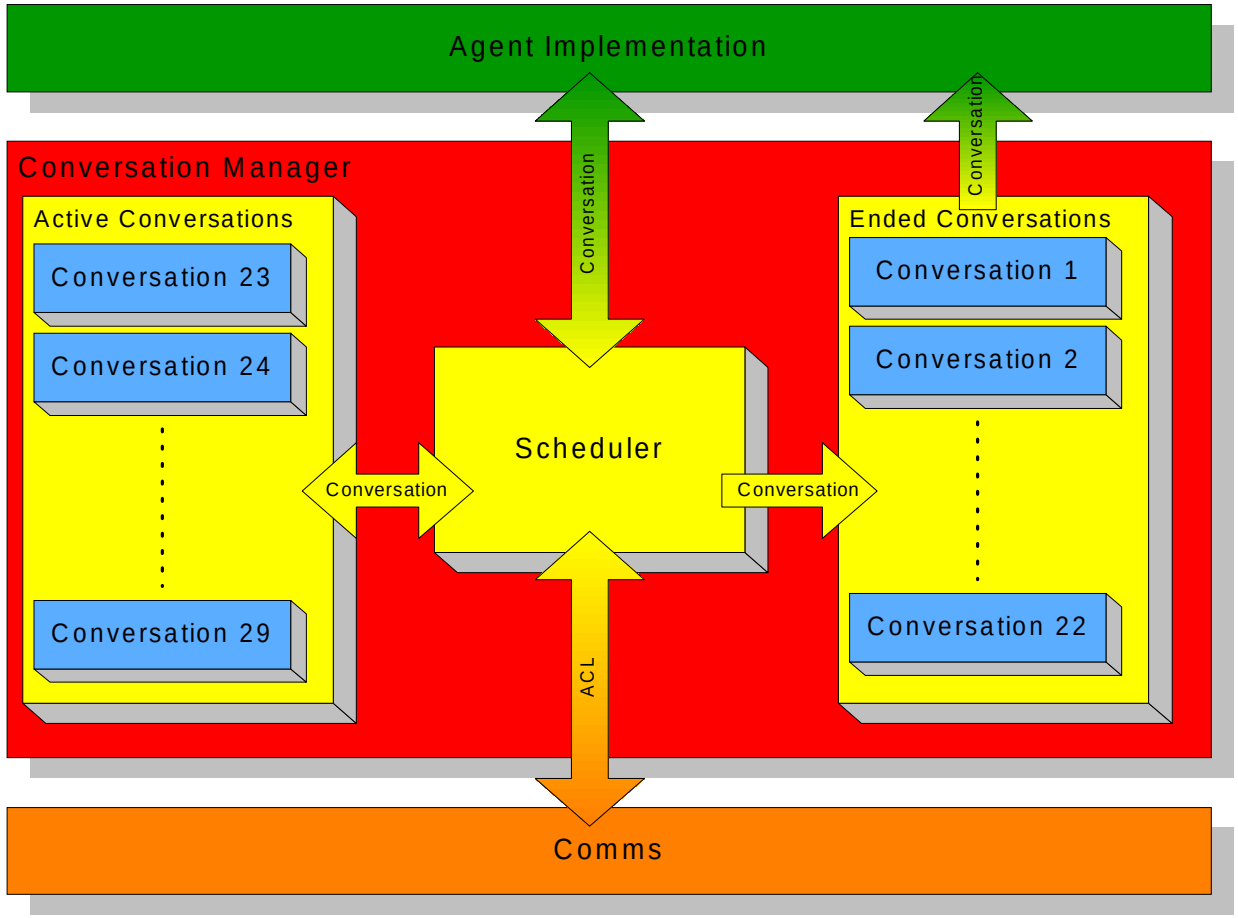
Conversation Problem

- Conversations between agents can be extremely complicated
  - many messages sent and being replied
  - how to know into what conversation they belong?
- Option 1
  - individual ACL messages, conversation-ids
  - implementation complicated

Option 2:
Conversation Management

- Deal with conversations, not individual ACL messages
- CM checks conversation ID's and works out if new message is part of an existing conversation or a new one
  - if old, add message into conversation to correct place in the protocol
  - if new, create a new conversation

Conversation Manager



Conversation Management

- **conversations-in-progress** list
  - adds new messages to the list
  - adds messages that are part of existing conversations to the correct object in the list
- All conversations and messages are part of a FIPA specific conversation

Planner Scheduler

- PS notifies the agent of new messages
  - routes "sent" messages out of the CM through agent comms to the appropriate agent
- **completed-conversation** list
  - PS removes finished conversation objects from conversations-in-progress to the list

Future: Parser Factory

- When agent receives a message the PF finds out what languages the message is in and dynamically loads the correct parser
- Parsed into a generic content object which is syntax neutral, retaining the original meaning of the message
- Developer deals only with contents

FIPA-OS class structure

- aw.agent
  - Mandatory components of an agent
  - required to run the platform
  - Agent, AgentWorldAgent
  - aw.agent.conversation: CoversationManager, Conversation, protocols...

FIPA-OS class structure

- aw.comms
  - communications and transport classes of the platform
  - required to run the platform
  - Comms, AgentComms
  - aw.rmi, aw.sunidl, aw.voyager3

FIPA-OS class structure

- aw.fipa
  - FIPA specific classes: classes that define or handle FIPA specific data or objects
  - required to run the platform
  - AgentGUID, FIPACONSTANTS

FIPA-OS class structure

- aw.ont
  - ontology specific classes - classes that belong to an ontology component
  - aw.ont and aw.ont.fipaman package (not other sub-packages) are required to run the platform
  - aw.ont.fipaman, aw.ont.profile

FIPA-OS class structure

- aw.parser
  - language specific parsers - components in sub-packages named by the language that they apply to
  - required to run the platform
  - ACLMessage
  - aw.parser.ACL, aw.parser.rdf, aw.parser.sl0

FIPA-OS class structure

- aw.platform
  - platform specific classes that are not agent components: platform agent classes
  - required to run the platform
  - AMS, DF, ACC

FIPA-OS class structure

- aw.skill
  - optional agent components that can be ‘plugged-in’ to the agent to provide it with a specific skill
  - not required to run the platform in general, although any platform agents that use specific skills will require those specific classes
  - memory database and serialization database

FIPA-OS class structure

- aw.tools
  - platform and agent level tools, not required for the use of the platform
    - testing harnesses, information tools etc.
- aw.util
  - helper classes (not application specific, not part of a skill)
  - required for correct operation of the platform
    - UTC time handlers, diagnostics output handlers

Example agent

```
import aw.parser.acl.ACLMessage;
```

- to be able to send and receive ACL messages

```
import aw.agent.AgentWorldAgent;
```

- all agents extend AgentWorldAgent

```
import aw.agent.conversation.*;
```

- send and receive conversations using CM

```
private static String _receiver;
```

- global variable

Main method

```
public class TestAgent extends AgentWorldAgent
{

public static void main( String[] args )
{
   TestAgent test = null;

   if ( ( args.length == 2 ) || ( args.length == 3 ) )
   {
        if ( args.length == 2 ) {
               test = new TestAgent(args[0], args[1],
                                    args[1], false);
        }
```

Main method

```
        else
        {
                _receiver = args[2];
                test = new TestAgent(args[0], args[1],
                                args[1], true);
        }
    }
    else
    {
        System.out.println("Usage: TestAgent <platform.profile
                        location> <agent-name> [<receiver-
                        agent-name>]” );
        }
    }
```

Constructor

```
public TestAgent(   String platform_profile_location,
                    String name,
                    String agent_owner,
                    boolean sender )
{
   super( platform_profile_location,
          name, agent_owner, false, true, true );
```

- AgentWorldAgent constructor, booleans: using push method, using conversation manager, write debug to disk

Constructor

```
    startPushing();
    if ( sender )
    {
        System.out.println( "This agent is going to start
                             sending messages to " +
                             _receiver );
        send( _receiver );
    }
    else
    {
        System.out.println("Agent " + this.getGUID() + " is
                             waiting to receive a message");
    }
}
```

Sending a test message

```
private void send( String agent_name )
{
   try
   {
       ACLMessage request = getNewConversation("fipa-
                            request").getFilledInMessage();
       request.setMessageType( "request" );
       request.setSender( this.getGUID() );
       request.setReceiver( agent_name );
       request.setContent( "(Hello)” );
       forward( request );
   } catch (UnknownProtocolException upe) {}
}
```

Receiving a message

```
Public void notify( conversation conv )
{
   ACLMessage acl = conv.getMessage
                    ( conv.getLatestMessageIndex() );
   If ( acl.getMessageType().Equals("request") )
   {
       ACLMessage agree = new conv.getFilledInMessage();
       […]
       Forward( agree );
       ACLMessage inform = new conv.getFilledInMessage();
       […]
       Forward( inform );
   }
```

Receiving a message

```
    if ( acl.getMessageType().equals("inform") )
    {
        send( acl.getSender() );
    }
}
```

Questions?

- Ask me!
- http://www.nortelnetworks.com/fipa-os
- agent@nortelnetworks.com
- mailing list for FIPA-OS developers:
  - to subscribe, or see the archive http://fipaos.listbot.com/

References

- **FIPA-OS Information** [online]. Nortel Networks 1999. Available at <http://www.nortelnetworks.com/fipa-os> [Accessed 26 February 2000]
- Hadingham, R. and Buckle, P., 1999. **FIPA-OS: FIPA Everywhere!** Presentation at FIPA meeting in Kawasaki, Japan in October 1999.
- Buckle, P., 2000. **Service Reservation** [online]. Presentation at FACTS workshop in Ipswich, UK in 28th of February 2000. Available at <http://www.labs.bt.com/profsoc/facts/workshop/index.htm> [Accessed March 13 2000]
- **FIPA-OS** [online]. Presentation at FACTS workshop in Ipswich, UK in 28th of February 2000. Available at <http://www.labs.bt.com/profsoc/facts/workshop/index.htm> [Accessed March 13 2000]

Further reading

- **FIPA Home Page** [online]. Foundation for Intelligent Physical Agents, 2000. Available at <http://www.fipa.org/> [Accessed 1 March 2000]
- **FACTS Home Page** [online]. ACTS, 2000. Available at <http://www.labs.bt.com/profsoc/facts/> [Accessed 1 March 2000]