KQML

A Language and Protocol for Knowledge and Information Sharing

or

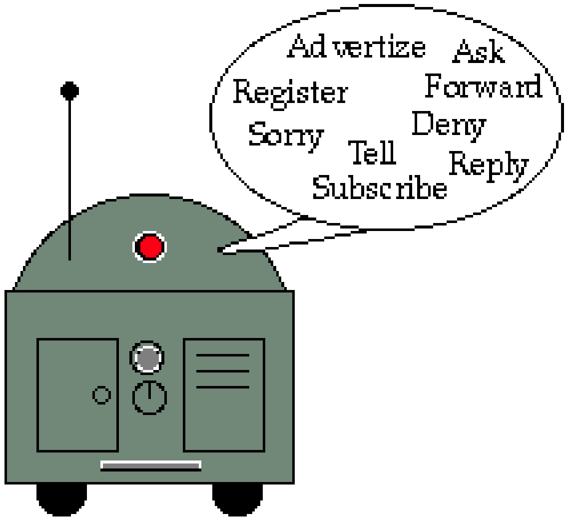
A Language for Knowbots

DAI Workshop
July 1994

Tim Finin & Rich Fritzson
University of Maryland
Baltimore County
Baltimore MD USA

Don McKay & Robin McEntire
Unisys Valley Forge Laboratory
Paoli PA USA

Overview

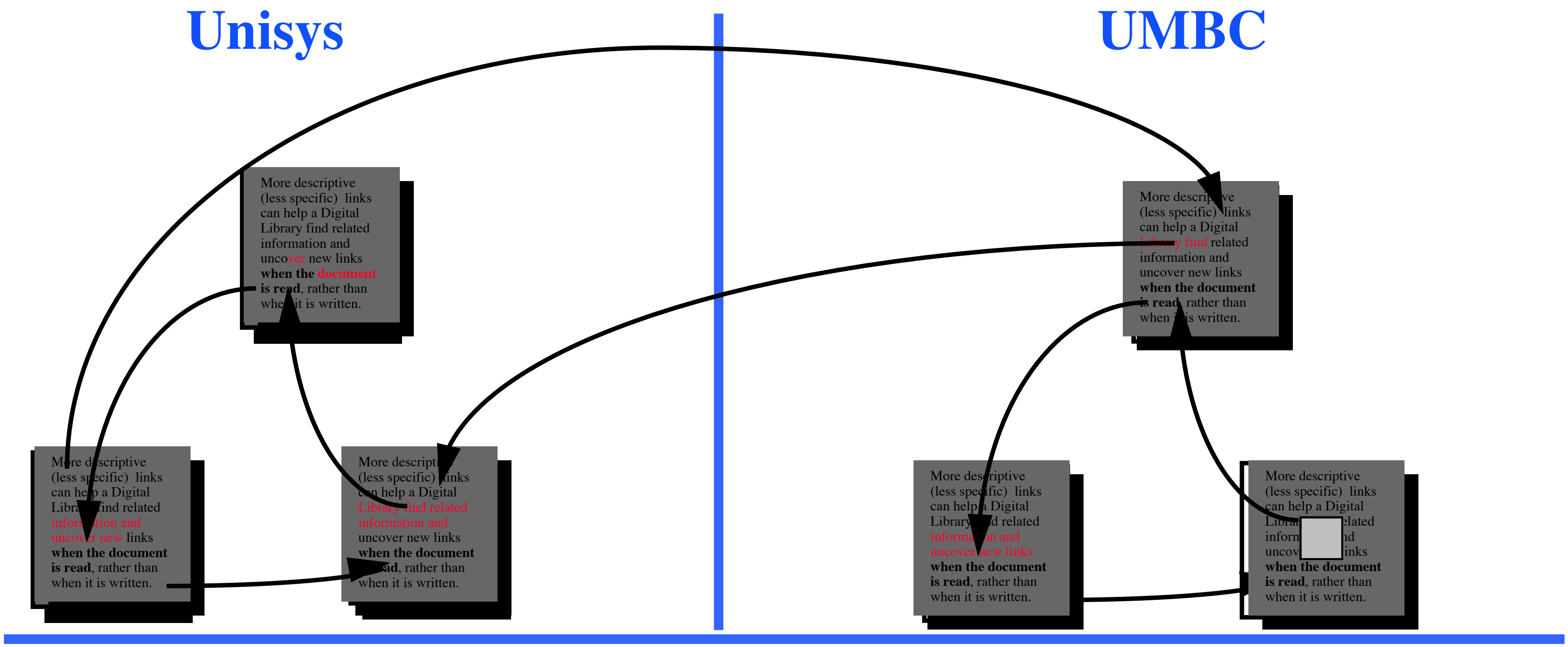


- The need for Intelligent Information Agents *(Knowbots)*
- The KQML language and protocol as the language for such agents
- KQML implementations, software architectures and applications
- Conclusions

Sponsors and Contributors

- Knowledge Sharing Effort
  - Advanced Research Projects Agency (G. Wiederhold, R. Neches)
  - Air Force Office of Scientific Research (A. Waksman)
  - National Science Foundation
  - Corporation for National Research Initiatives
- ARPA/Rome Laboratory Planning Initiative, BRL, ...
- Contributors --
  - at Unisys, Lockheed, EIT, ISX, etc.
  - at Stanford, Maryland, Toronto, USC/ISI, SUNY Buffalo, MIT, etc.

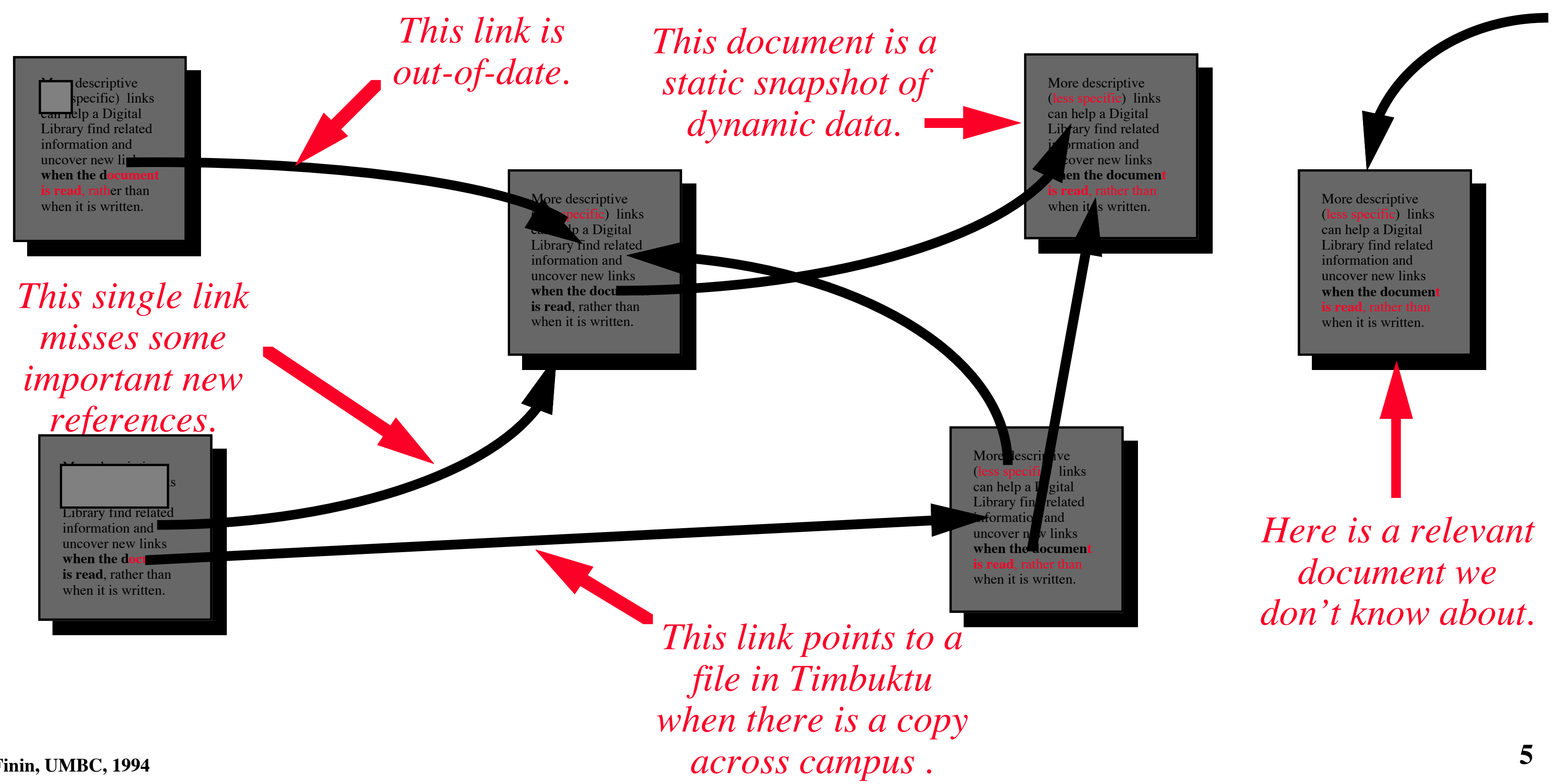
Motivation
World Wide Web



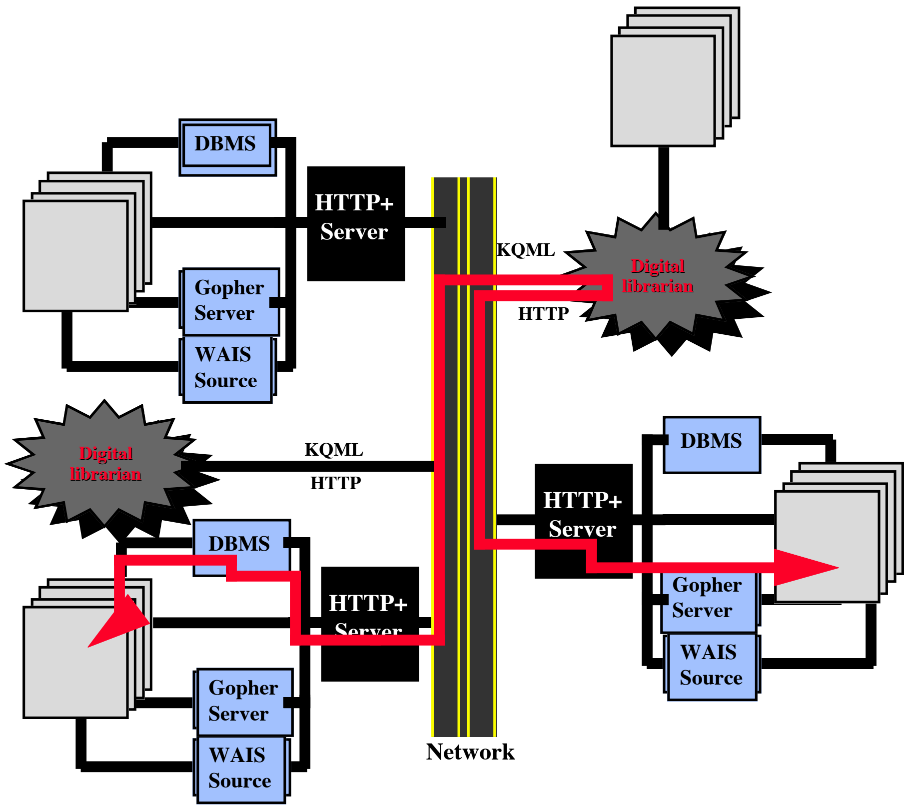
- **Distributed Hypertext uses links which can point to text on another hypertext system.**
- **HTTP is the WWW protocol.**

Improving the Web

Rigid, prewired links to static documents is not the right model



Motivation
Intelligent Web Agents



- **Links as queries**
- **Intelligent web agents will speak both KQML and HTTP, allowing them to:**
  - act as web clients speaking HTTP
  - act as web servers speaking HTTP
  - talk to each other□ and other agents using KQML
- **Thus they will co-exist and interoperate with**
  - current web clients (e.g., Mosaic, Lynx))
  - current□ web servers (e.g., httpd, gopher, etc.)
  - current DBMSs, expert systems and knowledge-based systems

ARPA Knowledge Sharing Effort

Developing techniques, methodologies and software tools for *knowledge sharing* and *knowledge reuse*.

Knowledge sharing can be viewed, in part, as *knowledge communication*.

ARPA Knowledge Sharing Effort
- Agent Communication Language -

For agent communication, a *Common Language* is needed:

Common Syntax

- **Interlingua Working Group**
- **Common Knowledge Representation Systems Specification (KRSS) Working Group**

Common Semantics

- **Sharable and Reusable KBs (SRKB) Working Group**

Common Pragmatics

- **Interfaces and Architectures Working Group**

Common Syntax

- A shared language for representing information.
- E.g., KIF, KRSL, the LOOM printed representation, object SQL, ...
- A universally accepted language is not likely soon.

➪ Interlingua Working Group developing KIF (Knowledge Interchange Language) based on *first order logic* with extensions.

Common Semantics

- Actually shared ontologies
- *Ontology* **:** A common vocabulary and agreed upon meanings to describe a subject domain.
- Relatively independent of syntax.
- Too difficult for just one. There will be many *domain specific ontologies* published.

⇨ SRKB working group developing tools and methodologies for building sharable ontologies.

Common Pragmatics

You must follow the same rules for discourse in order for others to find and speak with you.

- Who to talk to
- How to find them
- How to get their attention
- How to initiate and maintain a conversation
- How to deal with what they say
- How to respond appropriately

⇨ Interfaces and Architecture working group developing KQML.

KQML

- A high-level communication language and protocol for exchanging information independent of content syntax and ontology.
- A language in which to wrap information offering a uniform view of an *information agent as a knowledge base*.
- An extensible set of performatives expressing a *belief* or an *attitude* toward some information

➪ **The language of Knowbots**

A KQML Message

- Represents a single *speech act* or *performative*

ask	reply	forward	...
tell	deny	subscribe	
achieve	monitor	evaluate□	

- with an associated *semantics and protocol*
- with a list of *attribute/value pairs* , including

:content	:receiver	:aspect
:language	:reply-with	:in-reply-to
:topic	:label	...
:ontology	:from	
:sender	:to	

KQML Message Examples

```
(ask  :content (geoloc lax (?long ?lat))
      :language loom
      :ontology std_geo
      :from map_agent
      :to geo_server
      :label fm-query-10-23-92-14-09-07)
(tell :content "geoloc(lax, [42.45,83.21)"
      :language standard_prolog
      :ontology std_geo)
```

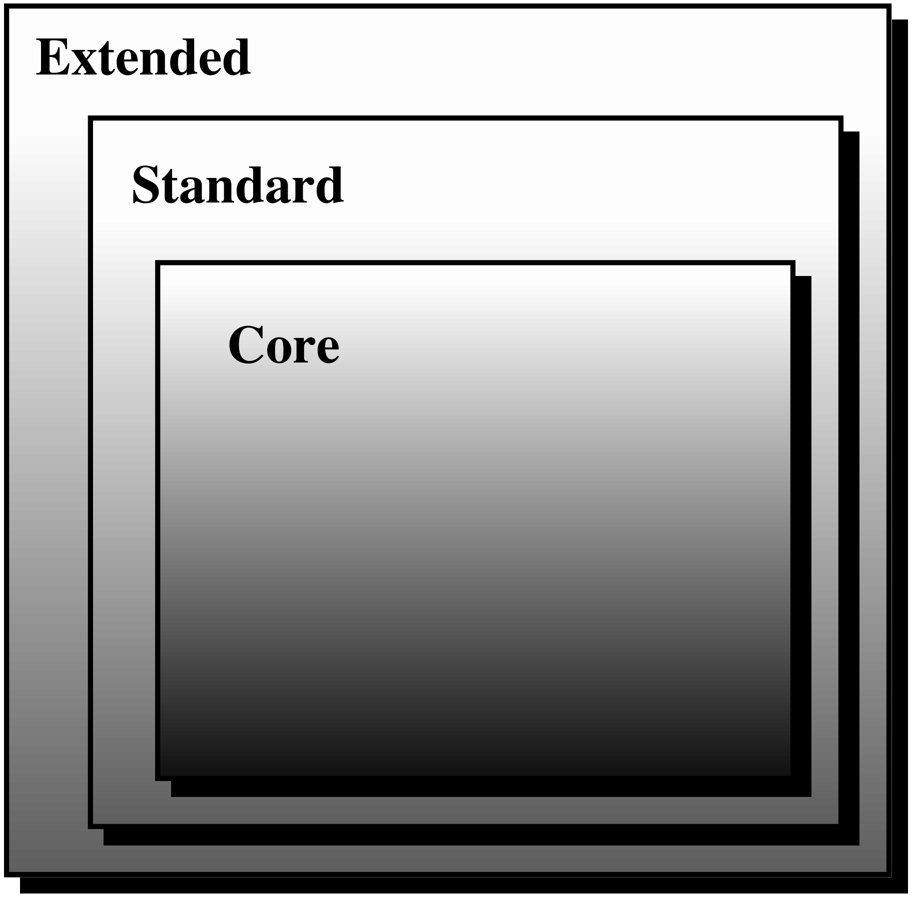
Performative

Parameter

Value

Current syntax is Lispy with provisions for binary data and structures.

KQML Performatives

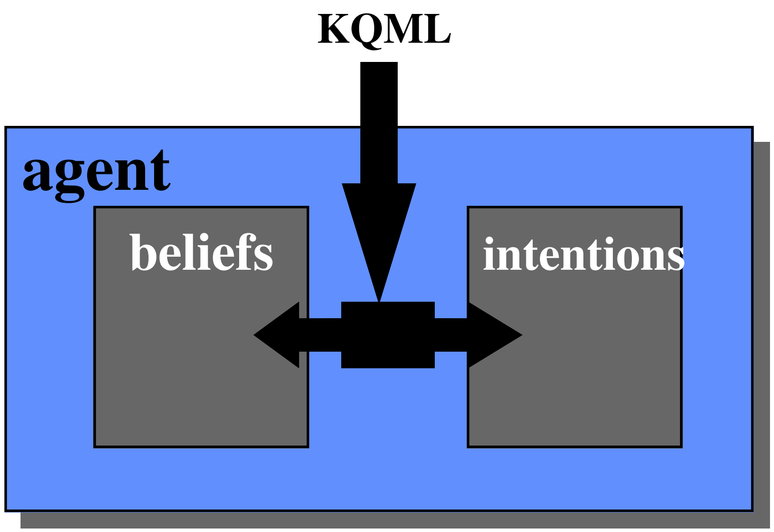


- **Core Performatives**
  A small set of primitive performatives formally defined which are used to define others.
- **Standard Performatives**
  Reserved names for performatives that most agents will want to handle.
- **Extended Performatives**
  Groups of agents can define and use extensions.

Semantics of Core Performatives

KQML agents are modeled as having

- an information store (beliefs)
- a goal store (intentions)



- **TELL** - add content to the information store.
- **UNTELL** - remove content from the information store.
- **ACHIEVE** - add content to the goal store.
- **UNACHIEVE** - remove content from the goal store.

Some KQML Performatives

- **Basic query performatives**:
  evaluate, ask-if, ask-in, ask-one, ask-all
- **Multi-response query performatives**:
  stream-in, stream-all
- **Response performatives**:
  reply, sorry
- **Generic informational performatives**:
  tell, achieve, cancel, untell, unachieve
- **Generator performatives**:
  standby, ready, next, rest, discard, generator
- **Capability-definition performatives**:
  advertise, subscribe, monitor, import, export
- **Networking performatives**:
  register, unregister, forward, broadcast, route

KQML Communication Protocols

KQML Protocols Include:

- simple synchronous transactions, e.g. query/reply, assert/ acknowledge
- multi-response performatives, e.g. generators.
- asynchronous replies, e.g. subscriptions, monitoring
- communication services, e.g. forwarding, broadcasting

Communication Level:

- packet oriented
- currently TCP/IP based, but has been carried by snmp (email), corba, http (www) and Linda.

KQML Supports Many Agent Architectures

Performatives for networking and for describing capabilities, interests and metadata will support many architectures, e.g.:

- **Agent Based Software Engineering**

  The PACT testbed uses *forward, distribute*, and other performatives to support problem decomposition in CAD/CAM.

- **Declarative Information Distribution**

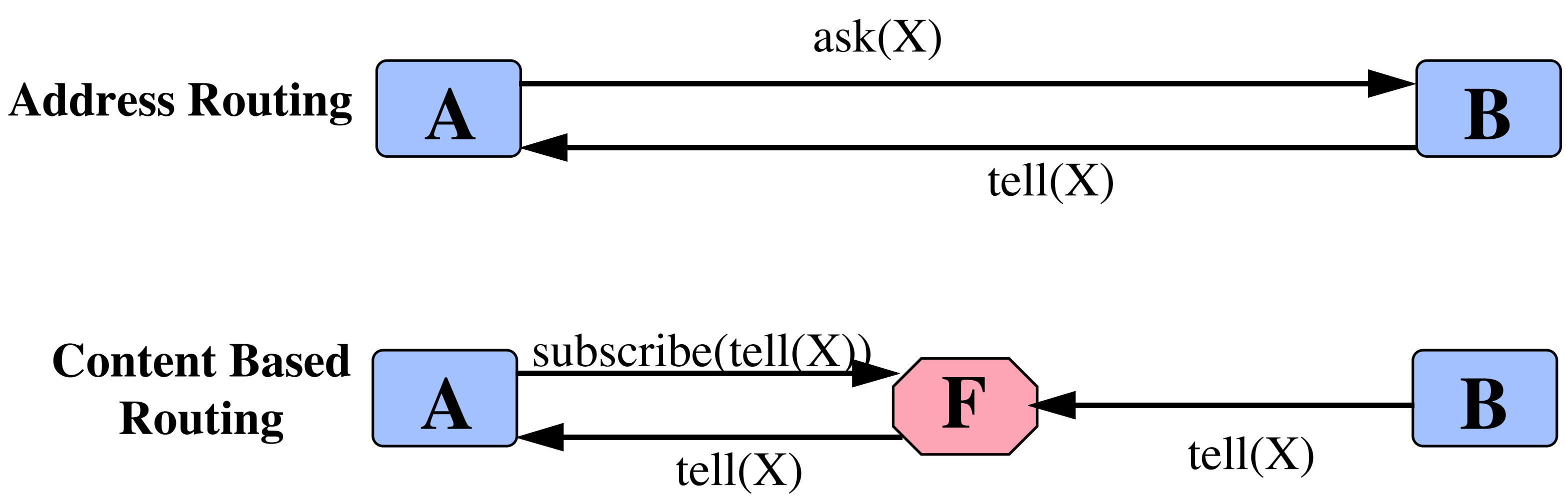
  Uses *routing agents* to support *declarative distribution* of information based on symbolic names of agents and declarations of topics.

- **Information Bus**

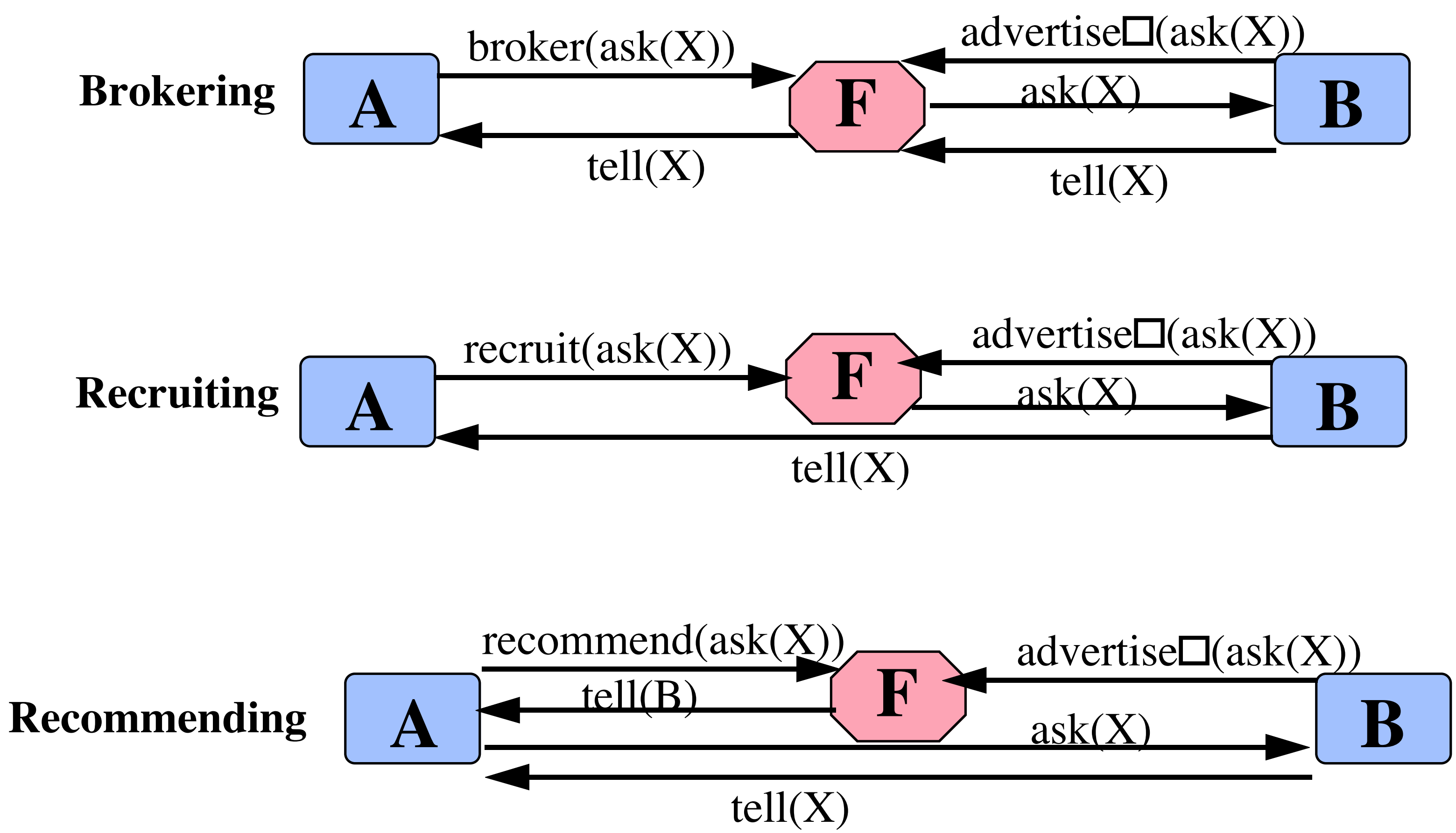
  Built on *publish and subscribe* primitives and the advertising of fixed hierarchical topic names.

Facilitation Services

- Facilitators are agents who traffic in meta-knowledge about other agents.
- Facilitators also provide resource discovery services□ and routing services.



Facilitation Services



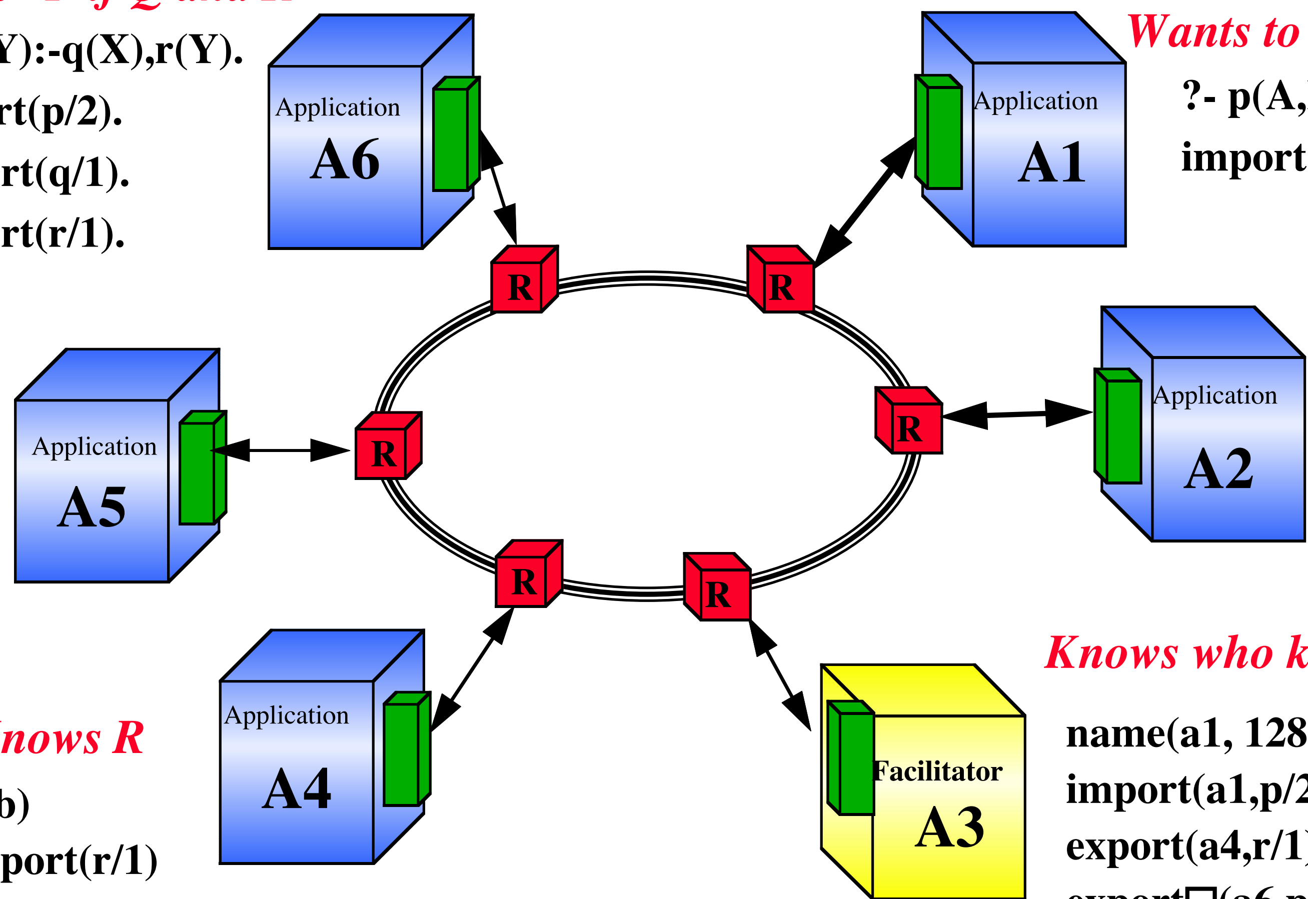
Content Based Routing

Knows "P if Q and R"

p(X,Y):-q(X),r(Y).
export(p/2).
import(q/1).
import(r/1).

Wants to know P

?- p(A,B)
import(p/2)



Knows Q

q(a)
export(q/1)

Knows R

r(b)
export(r/1)

Knows who knows what

name(a1, 128.62.39.4) . . .
import(a1,p/2)
export(a4,r/1)
export□(a6,p2) . . .

IMPLEMENTATIONS, □

SOFTWARE ARCHITECTURES

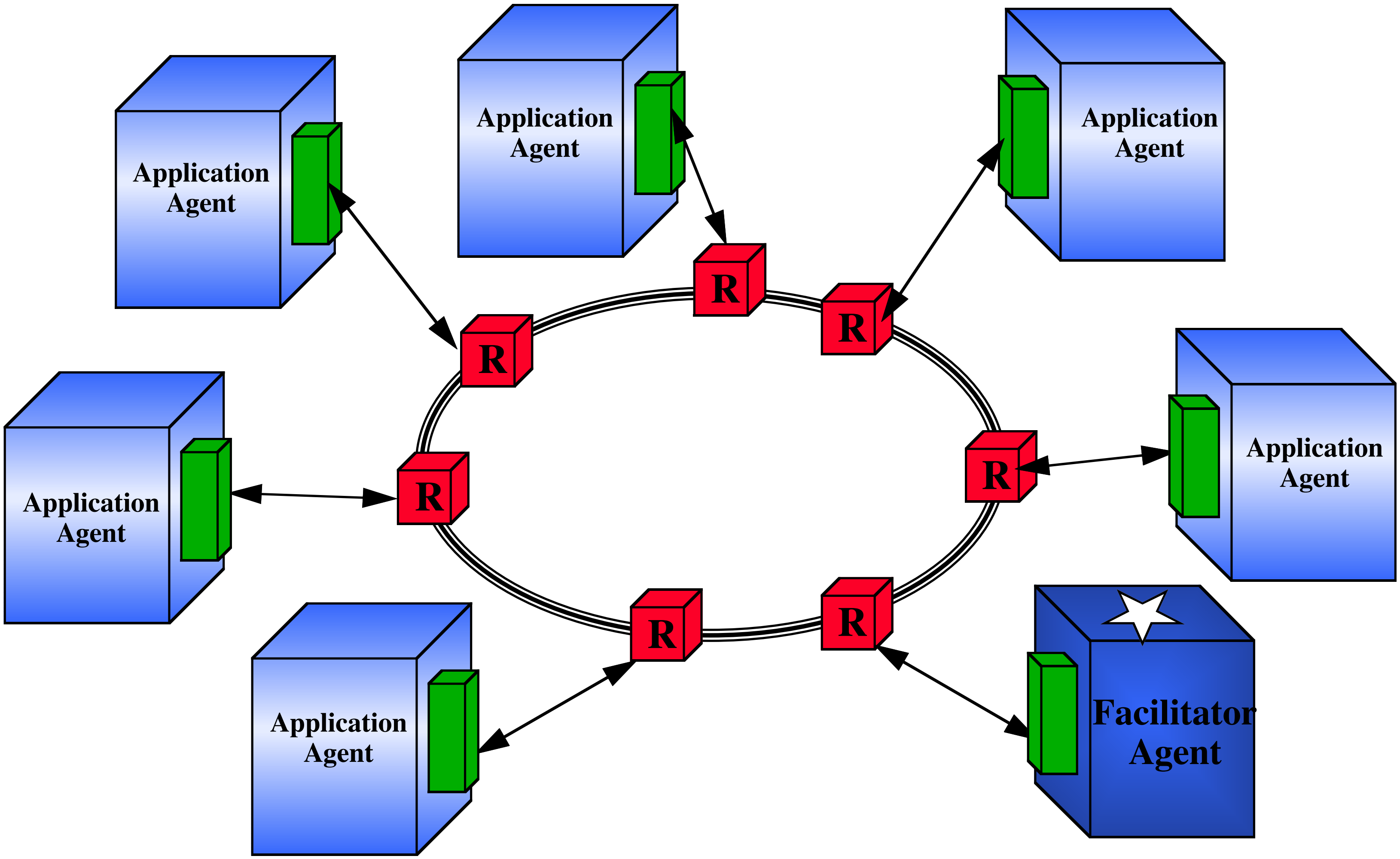
AND

APPLICATIONS

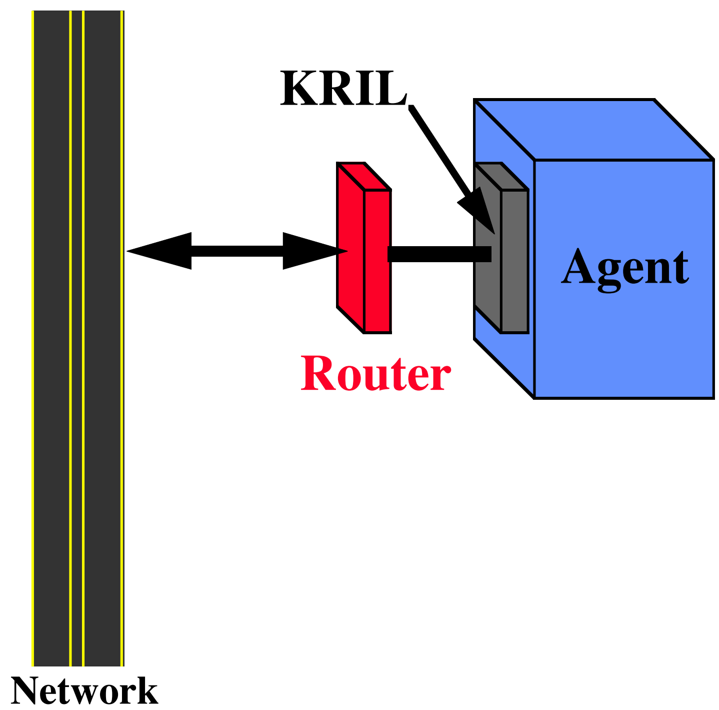
KQML APIs and System Interfaces

- KQML APIs
  - Unisys - Lisp and C
  - Lockheed/EIT/Stanford KSL - Lisp and C
  - Stanford Logic Group - Lisp
  - Toronto - Lisp and C
- System interfaces
  - LOOM and KRSL
  - Unisys Intelligent Database Interface
  - SIMS
  - Prolog
  - CLIPS

Unisys KQML Architecture

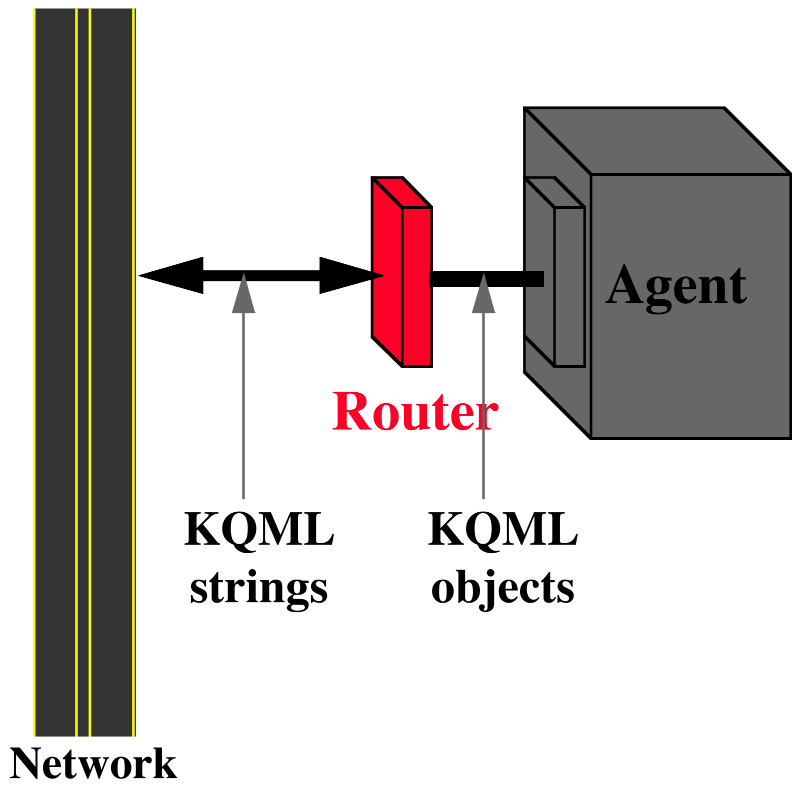


Unisys KQML Architecture



- **Routers**
  - Content independent message router.
  - One per agent - handles sending and asynchronous receiving of messages.
- **KRILs**
  - Knowledge Representation Interface Library.
  - API makes application look like a KB.
- **Facilitator Agents**
  - One per site - provides local DB of symbolically named services.
  - Eventually to hold and use a DB of topics and ontologies.
  - Can provide mediation and translation services.

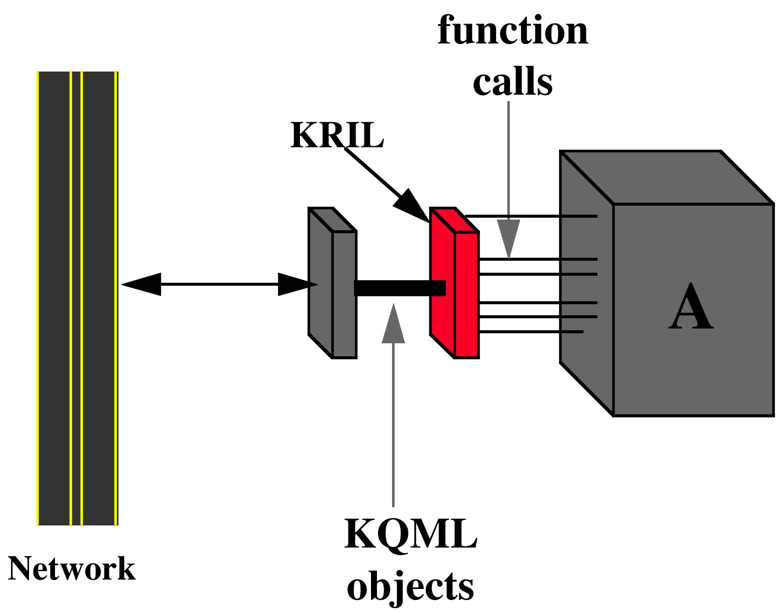
KQML Router



The Router provides an easy-to-use link between application and network.

- Provides both client and server capabilities
- Manages multiple simultaneous connections
- Handles some KQML interactions autonomously

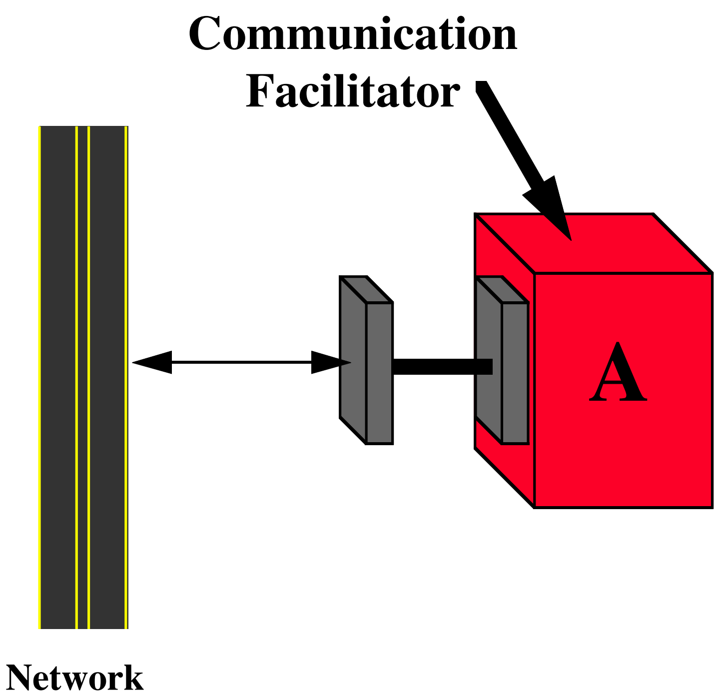
KQML KRIL



The KRIL is part of the application and has access to its internals.

- Provides internal access points to which the router delivers incoming messages.
- Analyzes outgoing messages for appropriate domain tagging and routing.
- Provides application specific interface and procedures for communication access.

Facilitator Agents



A Communication Facilitator is an agent that performs various useful services, e.g.

- Maintains a registry of service names.
- Forwards messages to named services.
- Routes messages based on content.
- Provides "matchmaking" between information providers & clients.
- Provides mediation and translation services.

KQML Developers and Users

- Unisys, ARPA/Rome *Planning Initiative, IWSDB*
- Palo Alto Collaboration Testbed (PACT)
- Lockheed AI Center, *SHADE, Shared Knowledge-Based Technology for the Re-engineering Problem*
- The Aerospace Corp., *WISARD - A Distributed Architecture for COTS Integration*
- Lockheed AI Center, *Cosmos, A System for Accumulating and Managing Distributed Design Knowledge*
- ISX, *IWSDB, Integrated Weapon Systems Database*
- Stanford University
- University of Maryland
- University of Toronto

KQML in the Future

- New performatives needed (e.g., security, negotiation?)
- Diverse development and user community.
- Implementation efforts are application driven.
- Investigate the use of KQML in NII applications, electronic commerce, digital libraries, concurrent engineering, healthcare inforation systems, ...
- Changes in the standards landscape: rehost KQML on OMG's CORBA, RDA, HTTP, etc ?

Suppose I want to use KQML?

- Add to your hotlist...
  - http://www.cs.umbc.edu/kqml/
- Join KQML mailing list
  - send mail to *majordomo@cs.umbc.edu* with text "subscribe kqml" in the body.
- Check out available APIs
  - several in Lisp and C, others in Prolog and CLIPS under constructio

Potential Problems

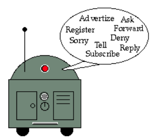
- Problems with exisiting APIs
  - not complete implementations
  - no off-the-shelf interfaces to system X (e.g., Loom)
- Problems with KQML
  - some features not fully articulated, e.g. security
  - potential design flaws or gaps?
- These are really opportunities to join the research effort
  - KQML is a need-driven, bottom-up solution

Summary and Conclusion

- The need for intelligent information agents (Knowbots)
- KQML as the language of Knowbots
- KQML supports many interesting agent architectures.
- KQML can be used for efficient communication
- Experimentation and experience will evolve the language

For More Information

- **Mailing lists**
  - kqml@cs.umbc.edu
  - kqml-developers@cs.umbc.edu
  - send email to *majordomo@cs.umbc.edu* with body text “info kqml” for more information



- **Information servers**
  - **www:** http://www.cs.umbc.edu/kqml/
  - **ftp:** ftp.cs.umbc.edu under pub/kqml
  - **gopher:** gopher.cs.umbc.edu under pub/kqml
- **Working group co-chairs**
  - Tim Finin -- finin@cs.umbc.edu
  - Jay Weber -- weber@eit.com