UNIVERSITY OF
Southampton

UNIVERSITY OF Southampton

# Open Hypermedia and the Web

COMP3220 Web Infrastructure

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

# What is Open Hypermedia?

An open system generally implies that:

- There is some interface by which third party programs may access the functionality of the system.
- The system may be accessed from applications on heterogeneous architectures.

Open hypermedia: hypertext features present within whole environment

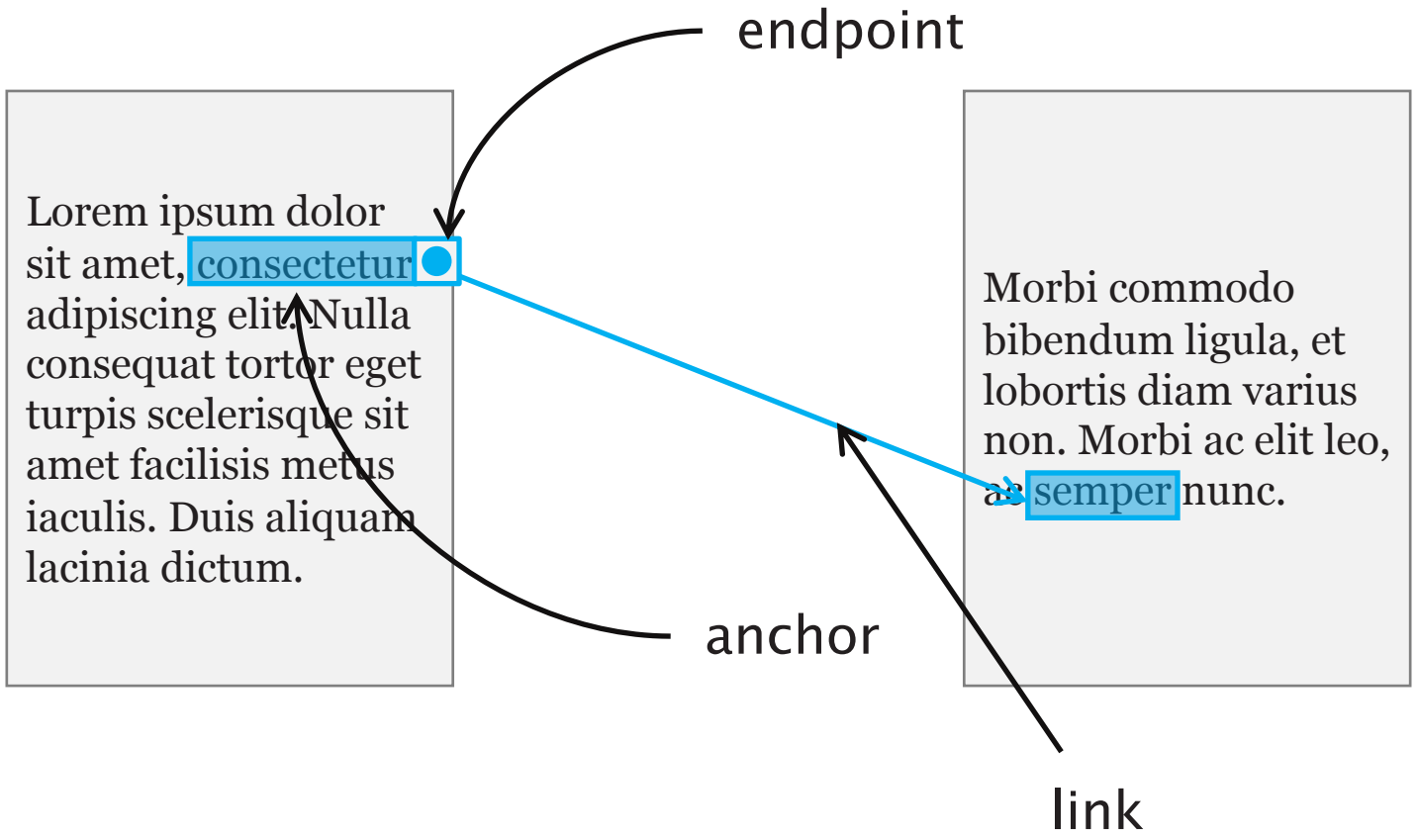
- Links and anchors are kept separately from documents (i.e. first-class links)
- Linkbases and link services

# What is Open Hypermedia?

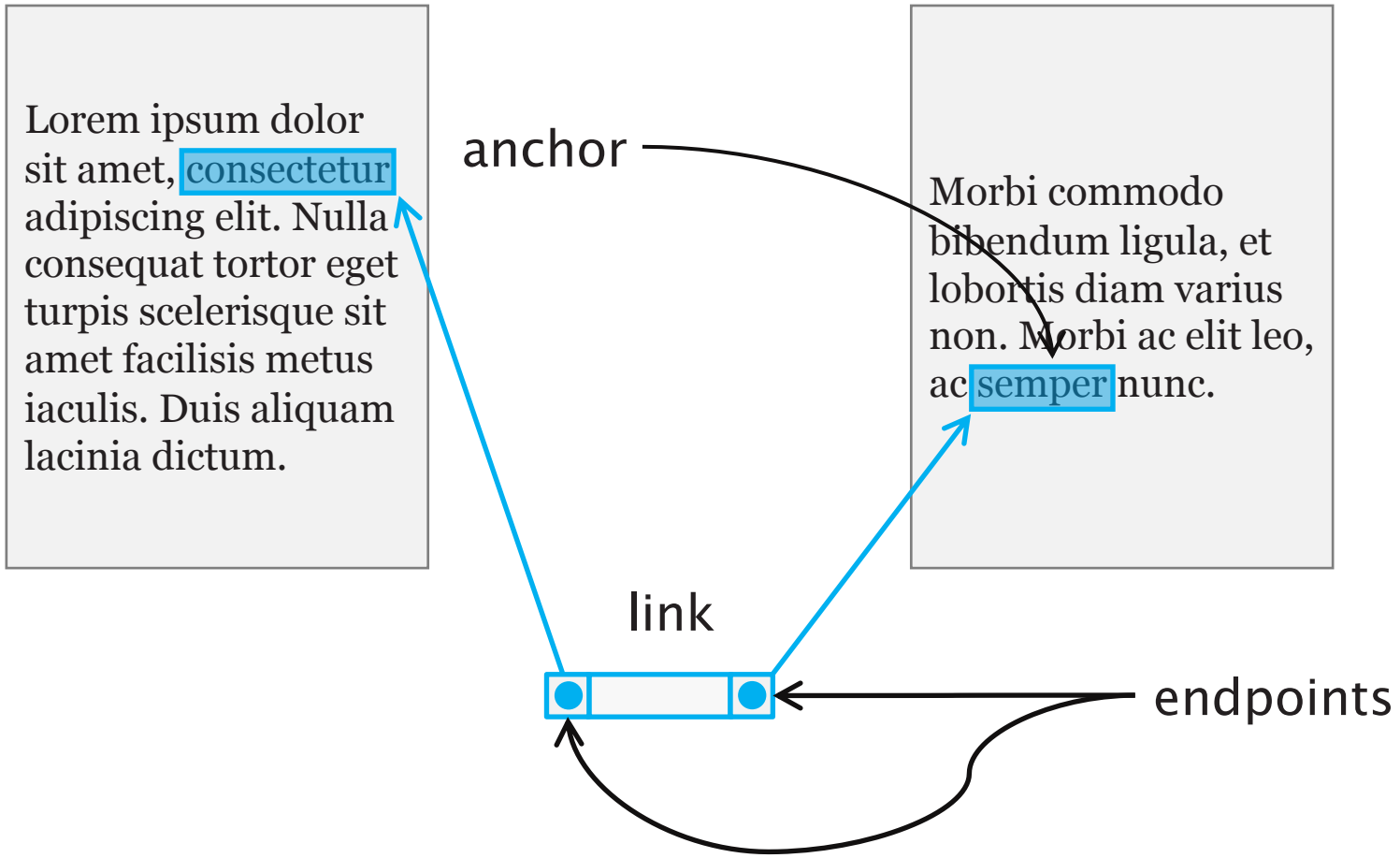
Open Hypermedia Systems are open with respect to:

- Applications
- Data formats
- Functionality (configurable and extensible)
- Other OHS systems
- Platforms
- Users

# Embedded links



# First-class links




# Dexter Hypertext Reference Model

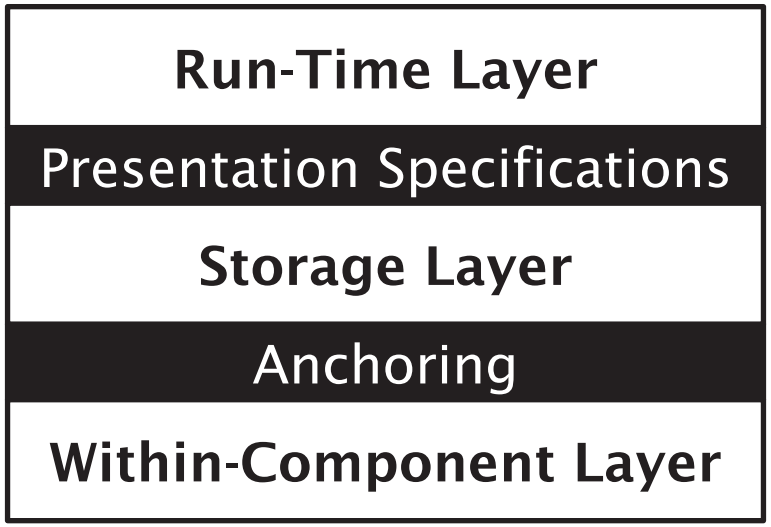
http://www.flickr.com/photos/hunkdujour/186391965/

# Dexter Hypertext Reference Model (1988-1990)

Formal model of an open hypertext system, not an implementation

Used to compare functionalities of existing systems

Used to design new systems and develop standards for interoperability




presentation, dynamics, user interaction

database of nodes and links

node content, structure within nodes

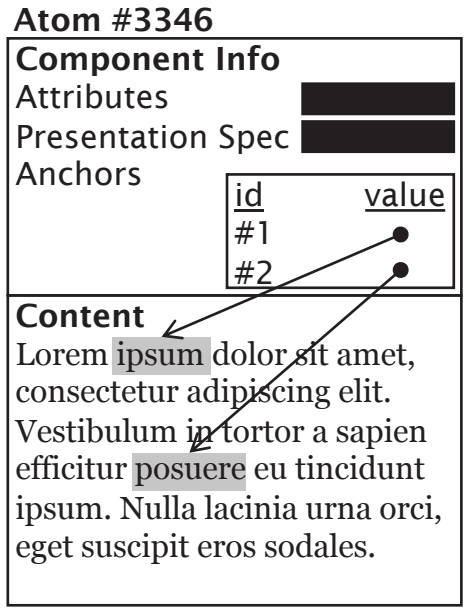
Halasz, F. and Schwartz, M. (1994) *The Dexter hypertext reference model*, Communications of the ACM, 37(2), pp. 30-39.

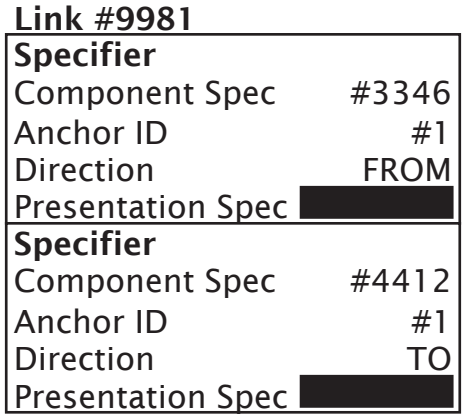
# Dexter storage layer

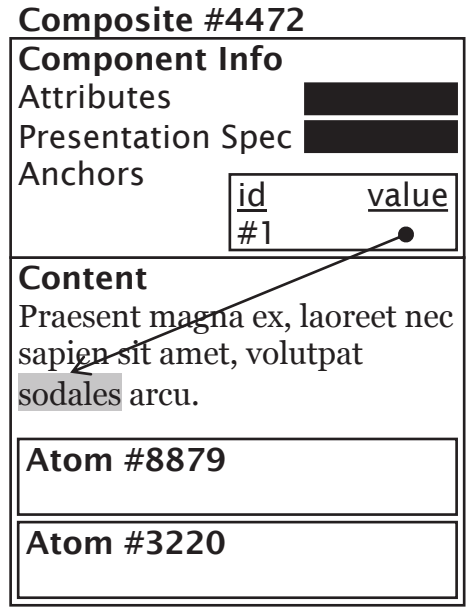
Three types of component: atoms, links and composites (sequences of components)

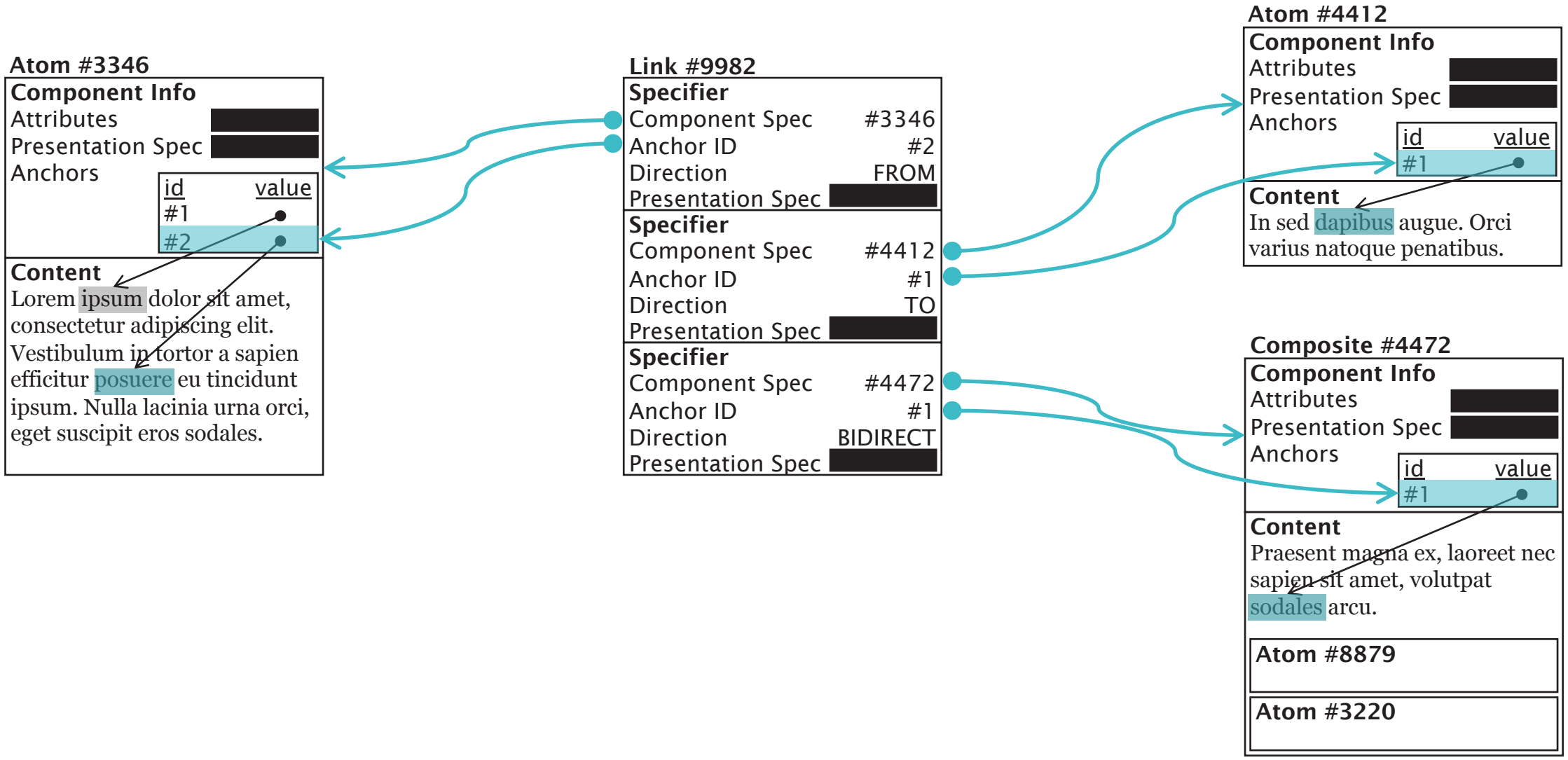
Key concept of a *specifier* – a robust description of a link endpoint

- Component specification, anchor, direction (to, from, bidirectional) and presentation spec
- Links as sequences of specifiers




Atom #3346
Component Info
Attributes
Presentation Spec
Anchors
id value
#1
#2
Content
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum in tortor a sapien efficitur posuere eu tincidunt ipsum. Nulla lacinia urna orci, eget suscipit eros sodales.
Link #9982
Specifier
Component Spec #3346
Anchor ID #2
Direction FROM
Presentation Spec
Specifier
Component Spec #4412
Anchor ID #1
Direction TO
Presentation Spec
Specifier
Component Spec #4472
Anchor ID #1
Direction BIDIRECT
Presentation Spec
Atom #4412
Component Info
Attributes
Presentation Spec
Anchors
id value
#1
Content
In sed dapibus augue. Orci varius natoque penatibus.
Composite #4472
Component Info
Attributes
Presentation Spec
Anchors
id value
#1
Content
Praesent magna ex, laoreet nec sapien sit amet, volutpat sodales arcu.
Atom #8879
Atom #3220

UNIVERSITY OF Southampton

# Hyper-G

# Hyper-G

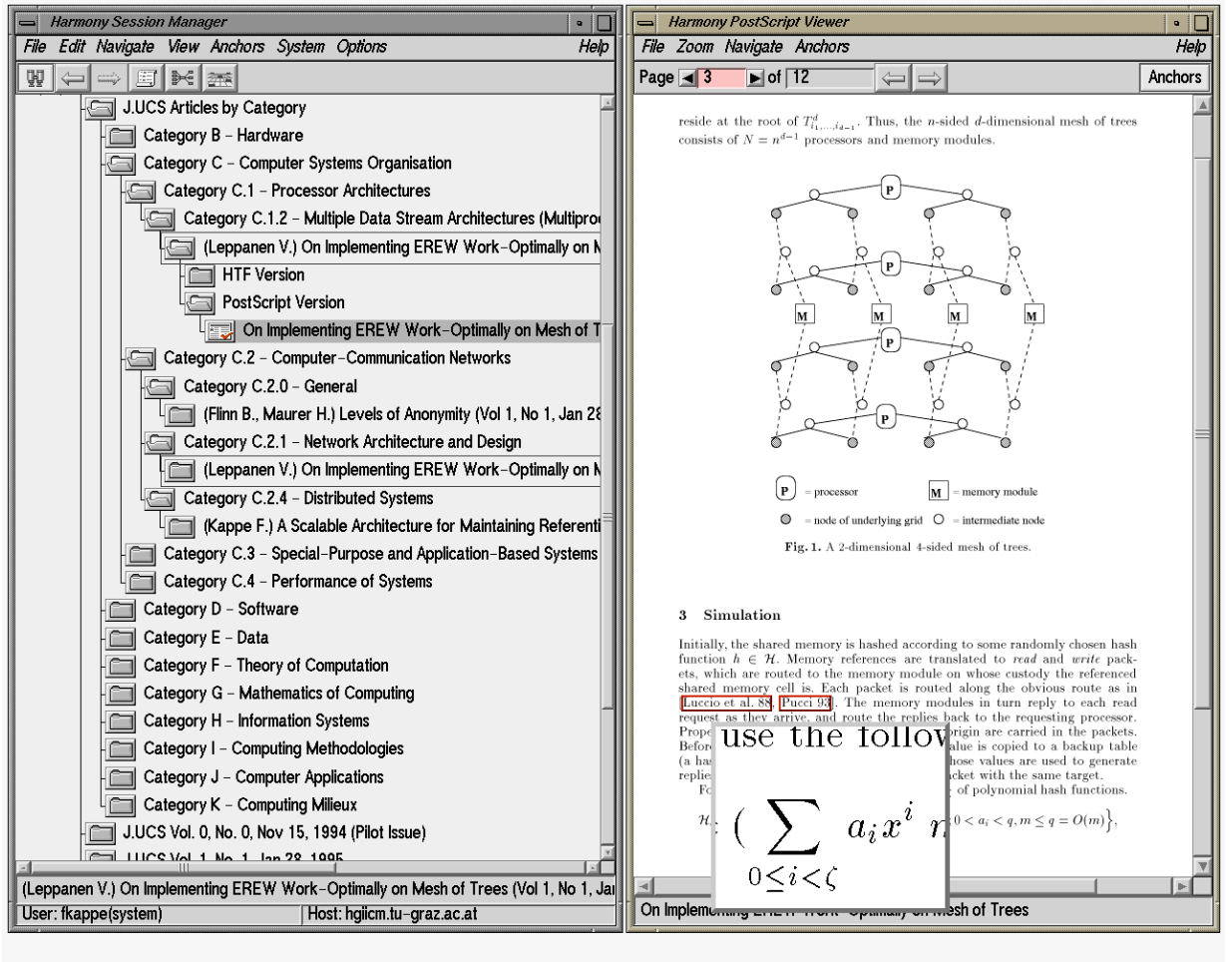
Developed at the Technical University of Graz, Austria around 1989-1990

Features:

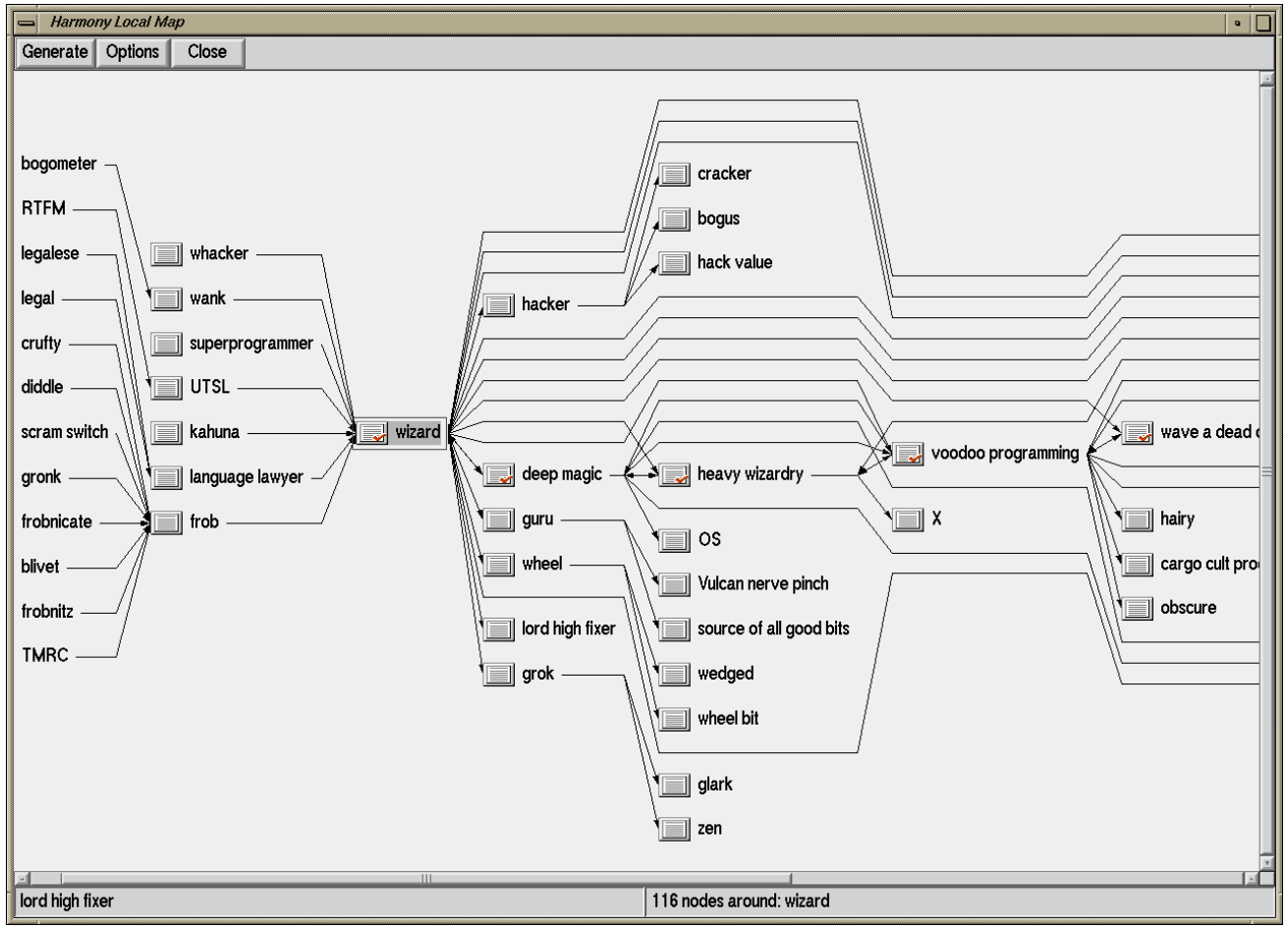
- Client-Server architecture (like the Web)
- Persistent session connections (unlike the Web?)
- First class nodes, links, anchors, composites and collections
- Bi-directional links
- Link integrity

Andrews, K., Kappe, F. and Maurer, H. (1995) *The Hyper-G Network Information System*, Journal of Universal Computer Science, 1(4).

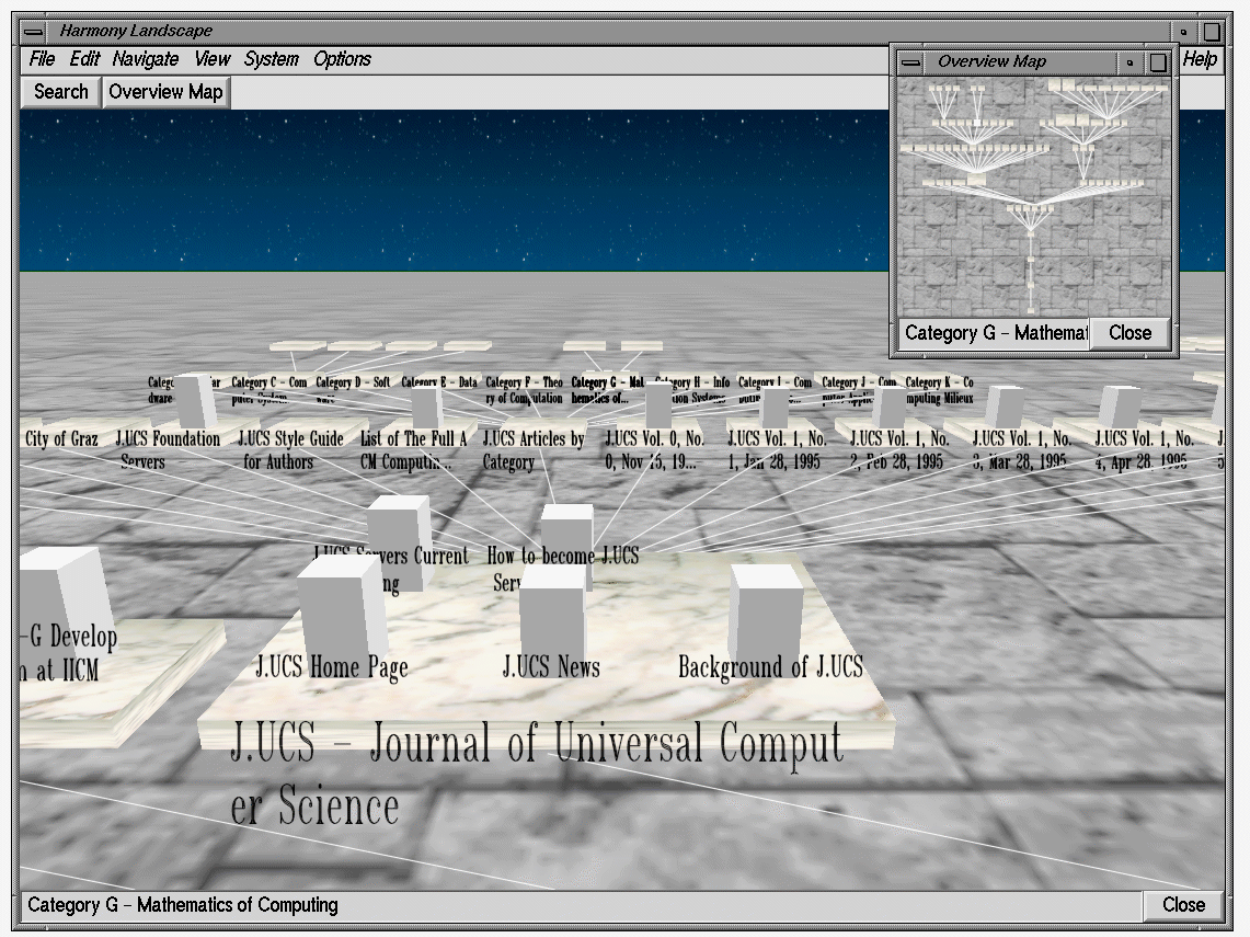
# Harmony client



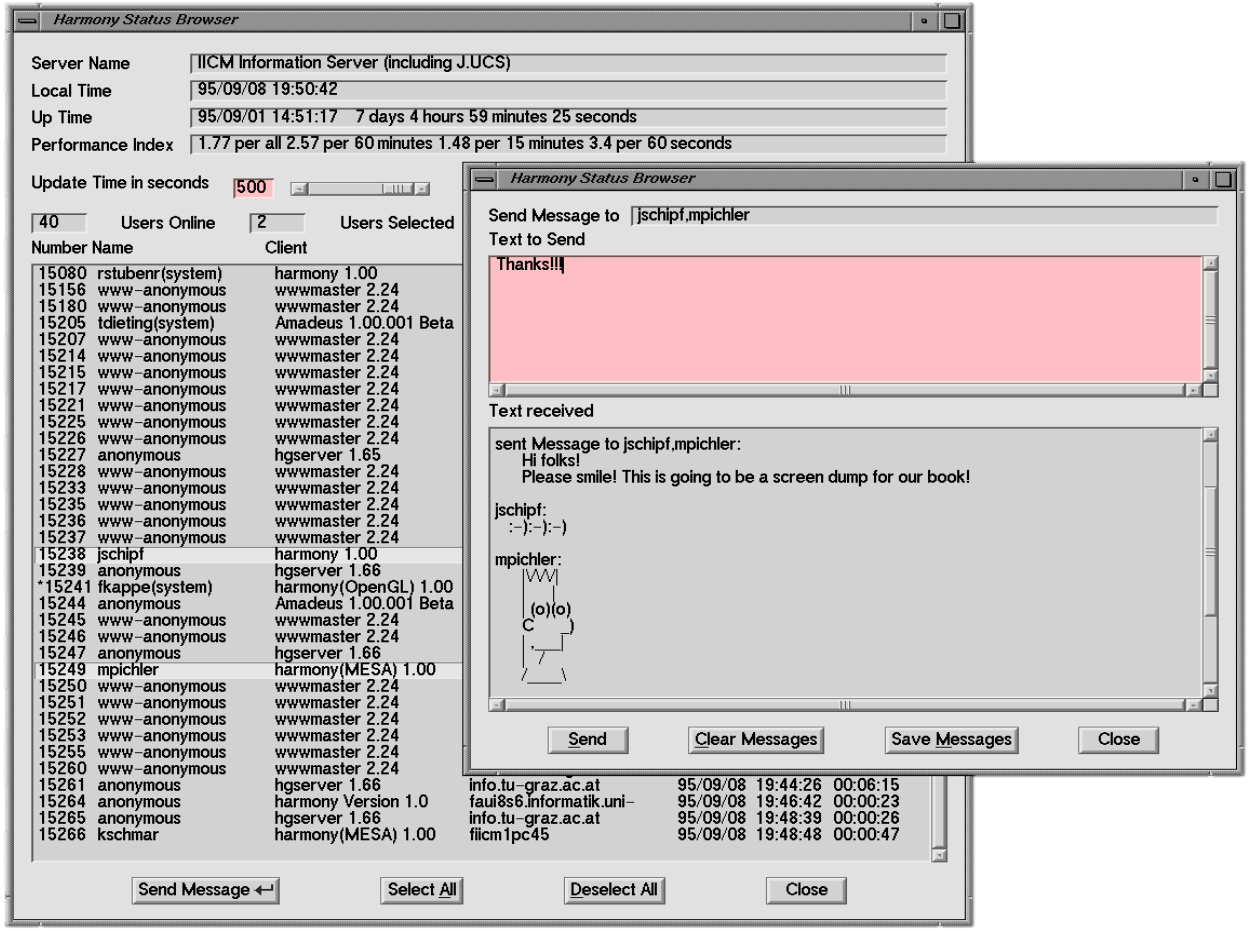
# Harmony client: link structure view



# Harmony client: landscape view



# Harmony client: client-client communications



# Link integrity

Users should always be able to follow any link that is presented to them

- When documents are moved, edited or deleted, links must be updated
- Compare with the Web: if the destination of a link goes away, the user sees 404 Not Found

Hyper-G divides links into two categories:

- *Core links* relate documents stored on the same server
- *Surface links* relate documents stored on different servers

Changes that affect core links can be processed by a single server

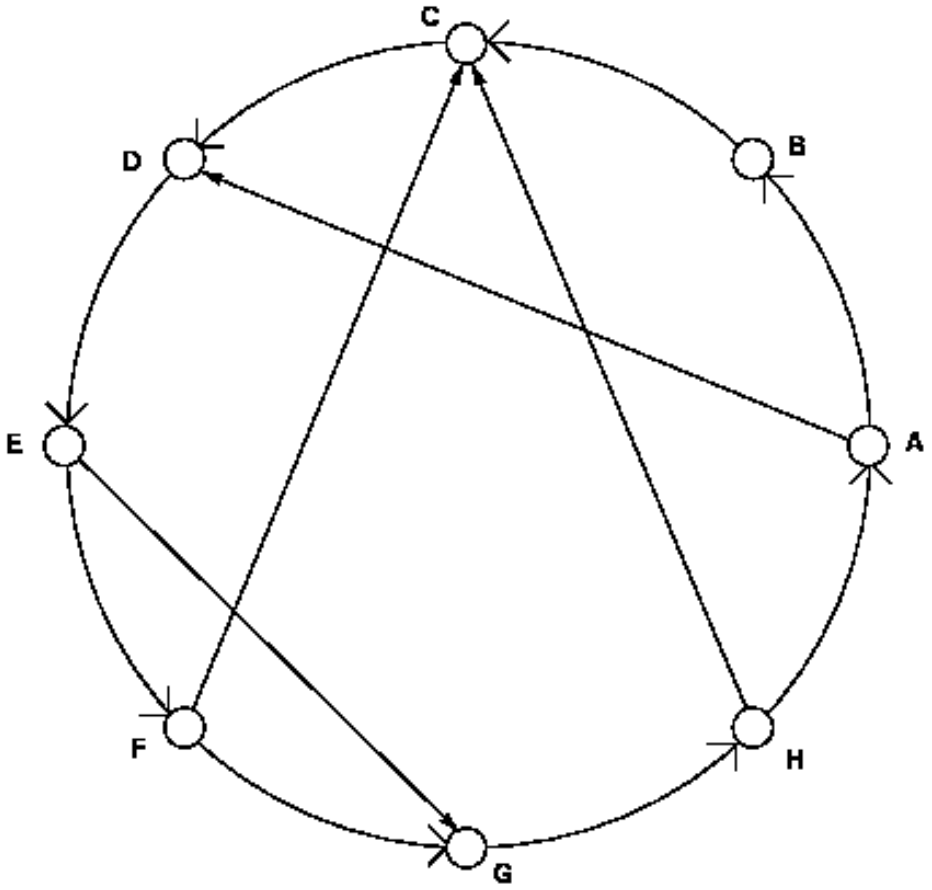
Changes that affect surface links involve multiple servers that need to be notified

# P-flood Algorithm

Broadcasting surface link updates to all affected servers is expensive

Probabilistic flooding algorithm used to send surface link updates more efficiently

- Servers arranged in a ring
- Servers send link updates to their immediate successor and randomly to other servers
- Scalable and robust
- Parameterisable – number of additional servers can be altered



# Evaluation

## Advantages

- Authoring support integrated into browser, and designed into protocols from outset
- Support for collaboration
- Early support for multimedia

## Disadvantages

- Own internet protocol (HG-CSP)
- Own markup language (HTF)
- Own browser (and other tools) (Harmony) - though simple browsing through a web browser too

- Flooding algorithm would probably not scale to the Web
  (but some similarities with peer-to-peer approaches like distributed hash tables)

UNIVERSITY OF Southampton

# Microcosm

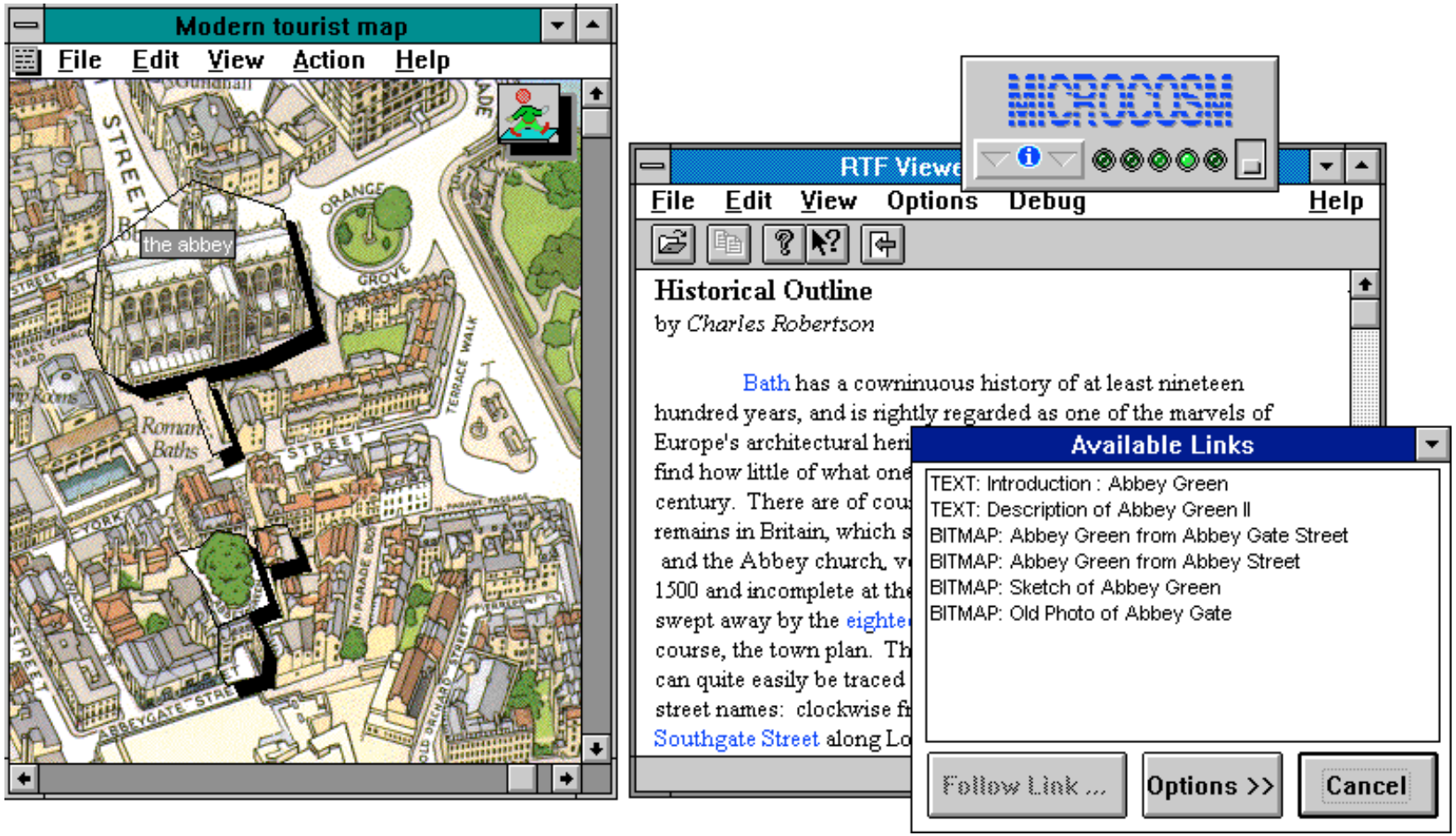
# Overview

Developed at the University of Southampton around 1989

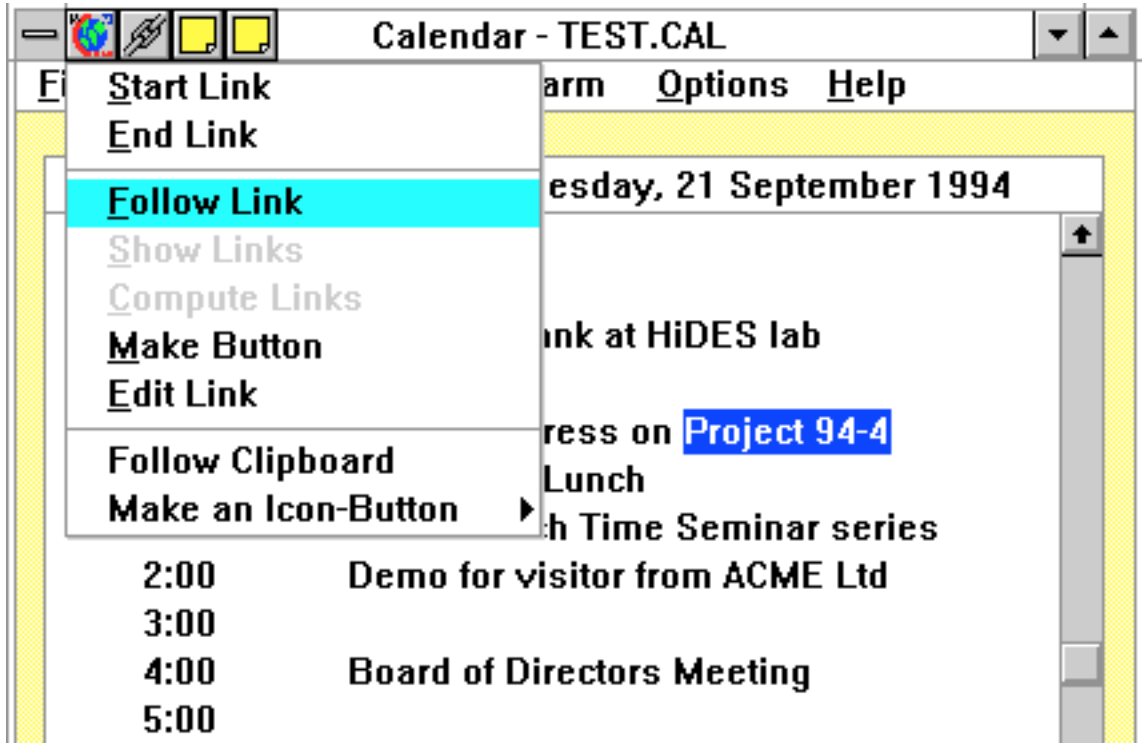
- Originally designed for use with read-only media (laser discs and later CD-ROMs)
- Originally designed as a desktop-based system, later expanded to a distributed system
- Informed much subsequent work within ECS: DLS, COHSE, etc

Fountain, A., Hall, W., Heath, I. and Davis, H. (1990). MICROCOSM: an open model for hypermedia with dynamic linking. Proceedings of ECHT90. pp. 298-311.

# Microcosm client

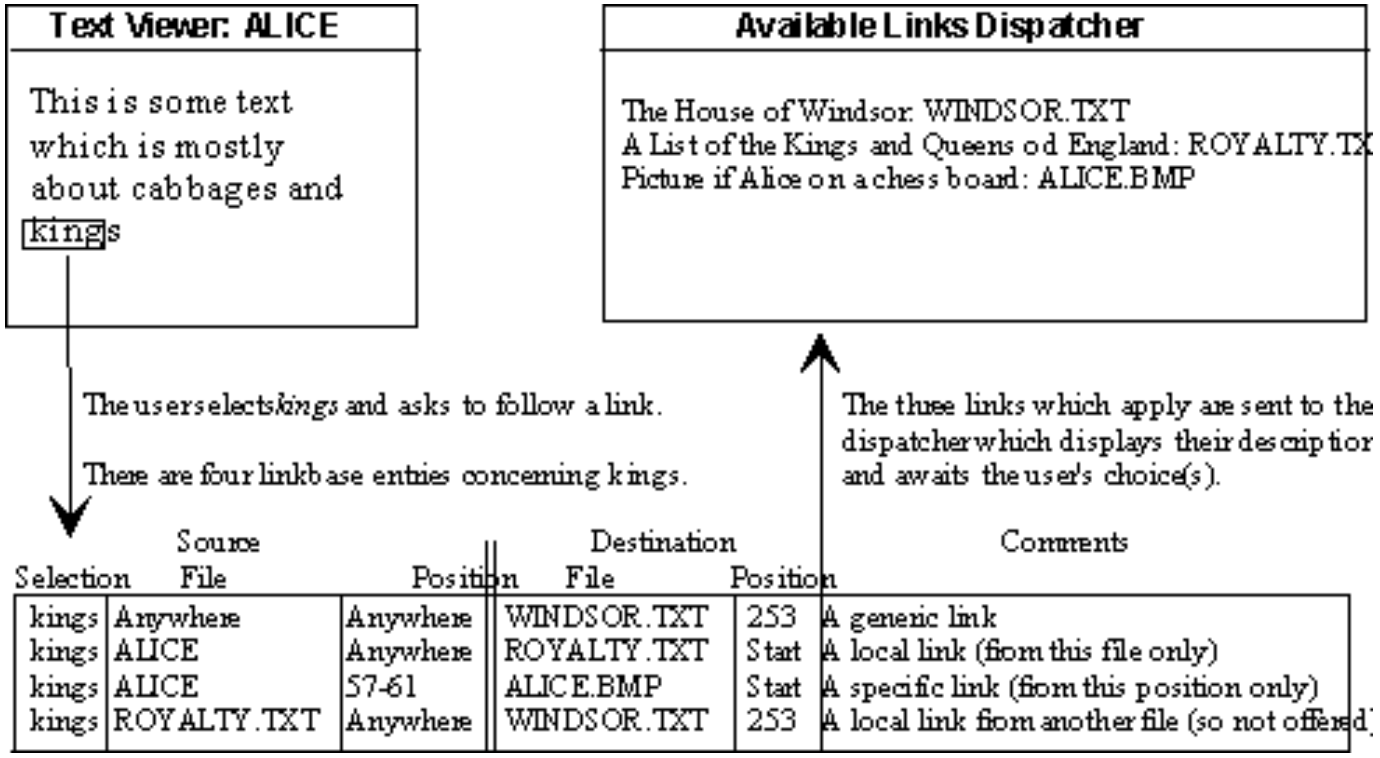


# Microcosm universal viewer



# Specific, local and generic Links

Position of source anchor held in the linkbase

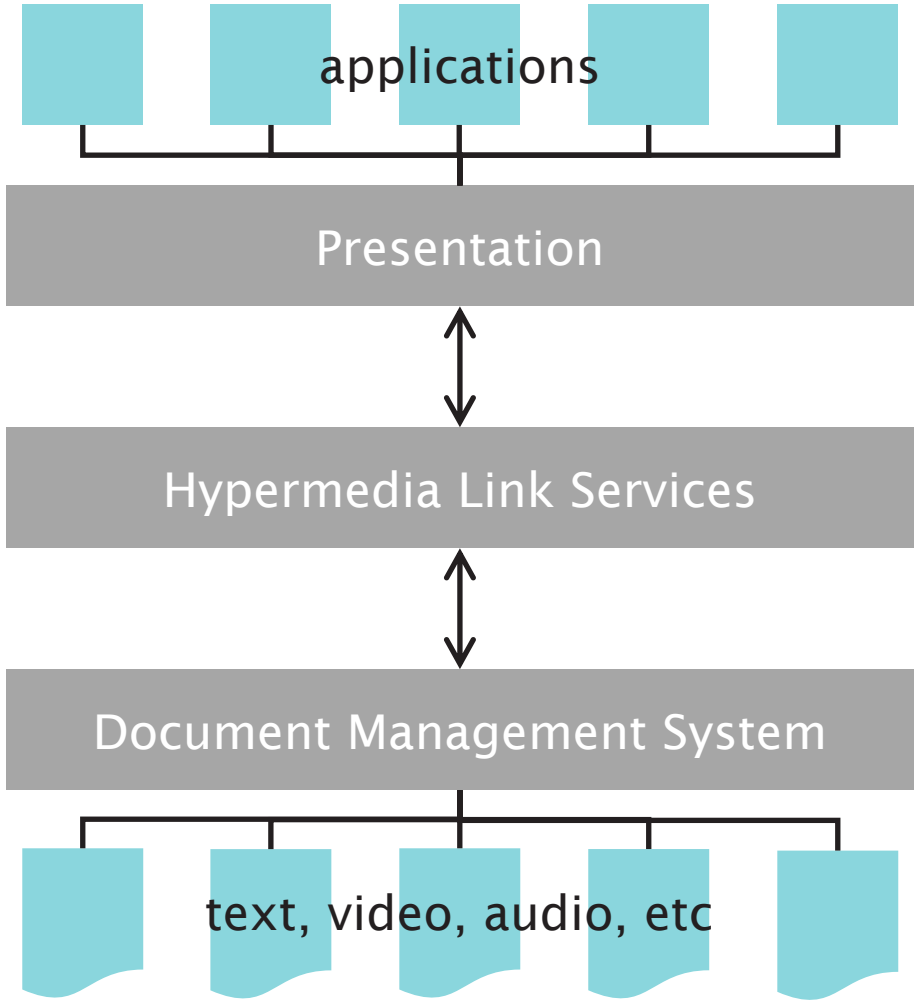


# Microcosm architecture

Separation of concerns:

- Document storage and management
- Link storage and management
- Applications and presentation

Link services mediate the other components

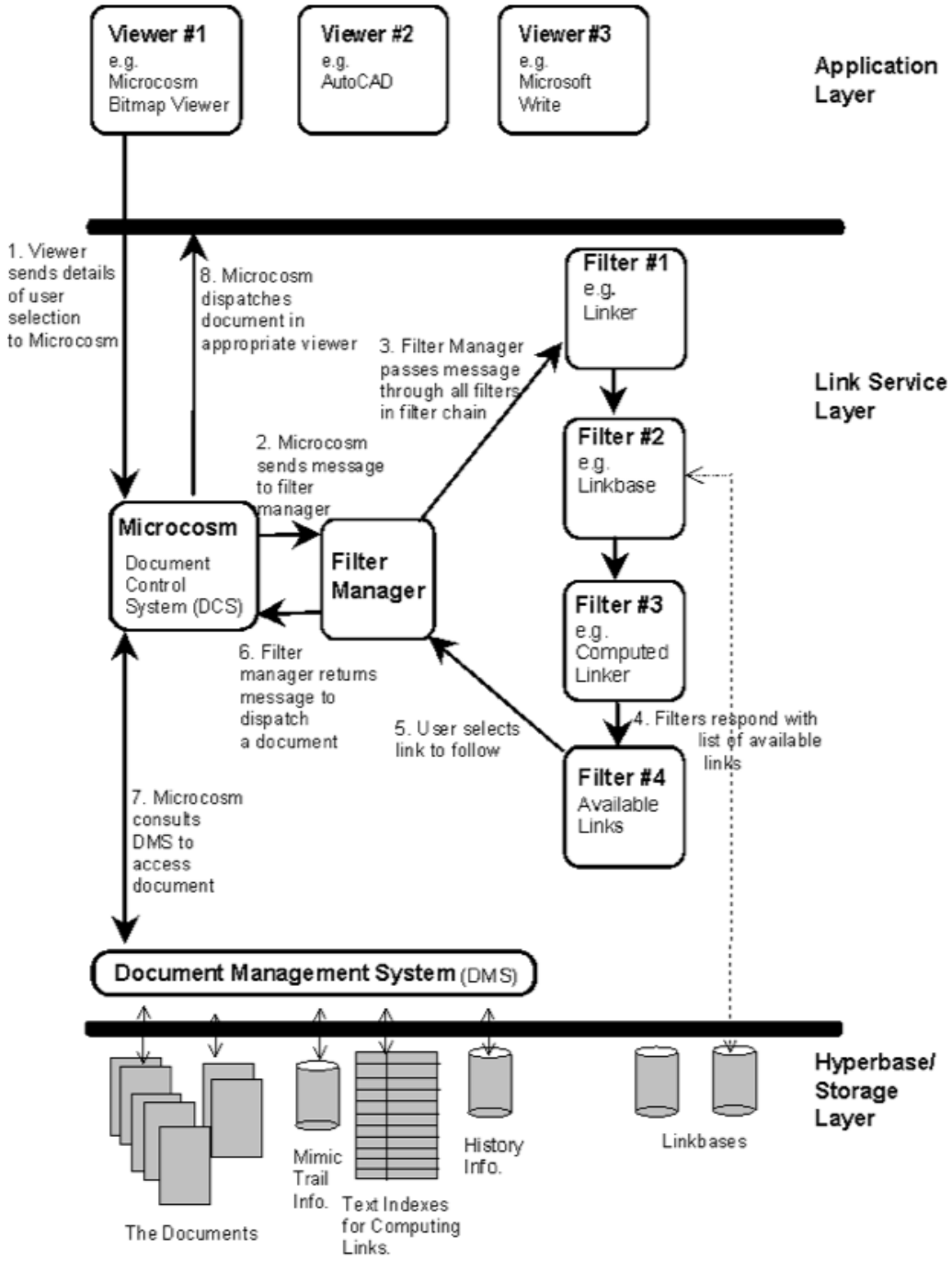


# Microcosm architecture

Link service comprises a sequence of *filters*

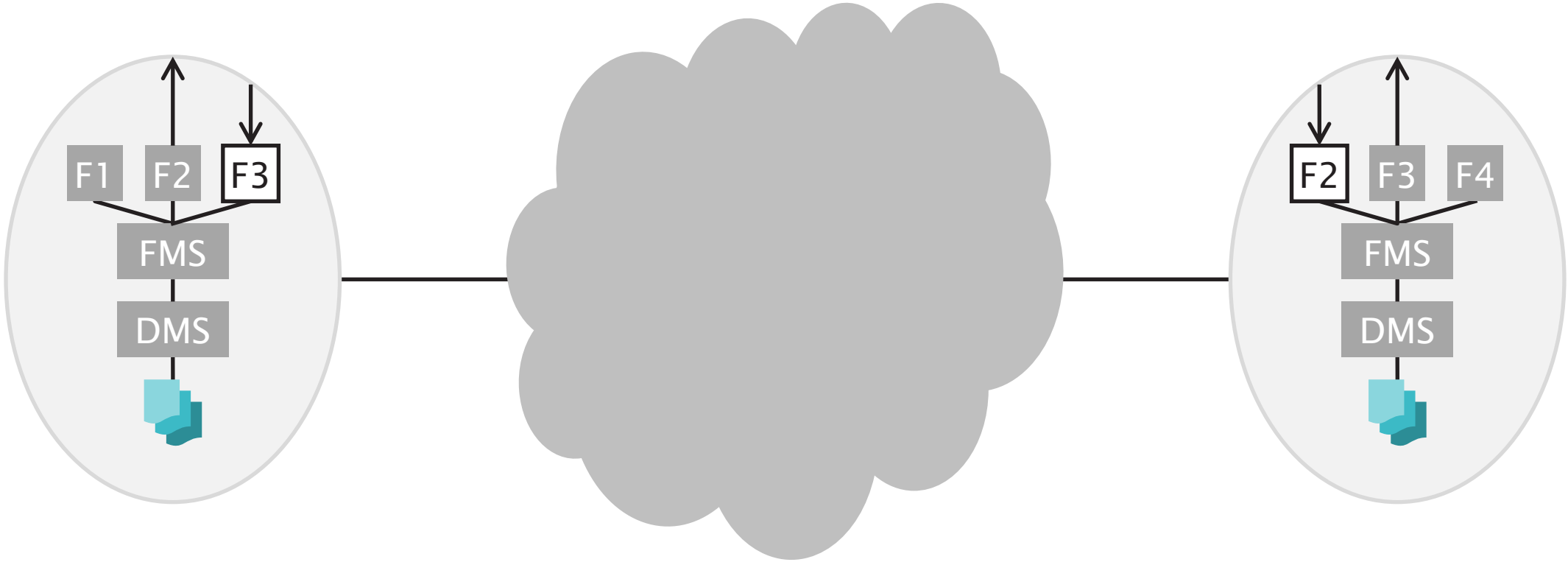
Filters are composable services that generate or manipulate links

*Filter manager* organises filters into chains



# Distribution

Microcosm instances may publish filters to be used by other instances



# Evaluation

## Advantages

- Rich model of linking
  (generic links, n-ary links, etc)
- Flexible document processing
  (multiple linkbases)
- Integration with third-party applications

## Disadvantages

- Poor scalability
  Intended for workgroups to enterprises
- Distribution not intended from the outset
- Arguably, no native document format
- No support for link integrity

UNIVERSITY OF Southampton

# Open Hypermedia Protocol

# Open Hypermedia Protocol

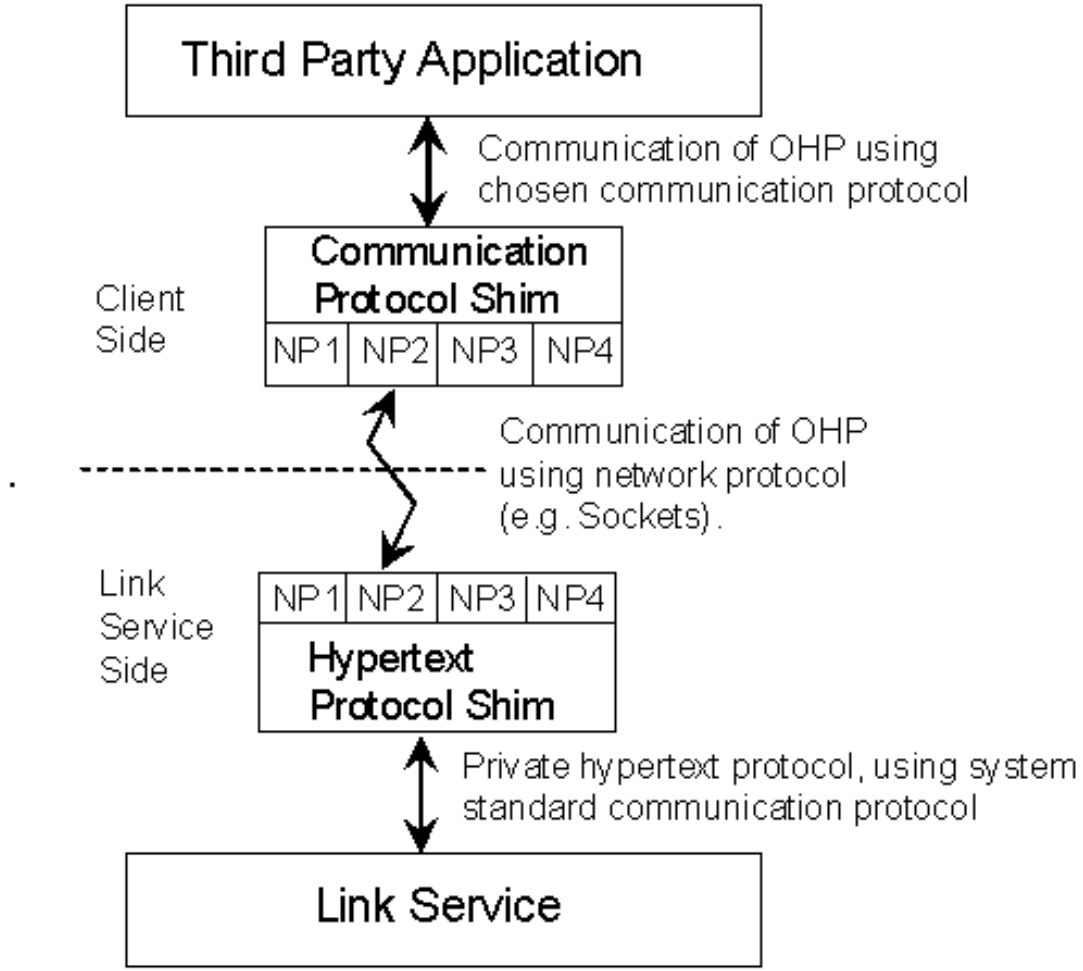
Initially a naïve attempt to “shim” existing linkservers so that they could be used by standard client integrations

- An early realization by the community that writing clients is far harder (and less interesting) than writing servers

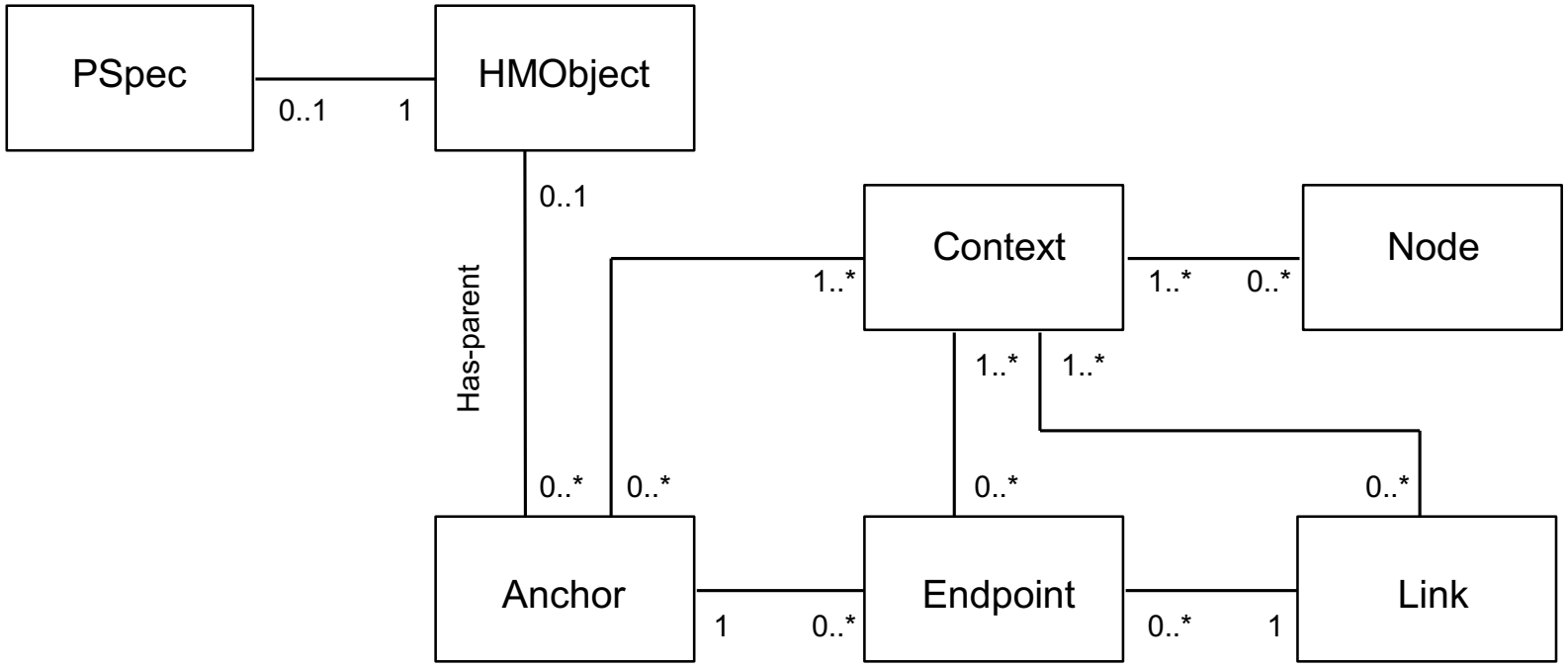
OHP formed the basis for later integration efforts (FOHM)

Reich, S., Will, U.K., Nürnberg, P., Davis, H.C., Grønbæk, K, Millard, D.E. and Haake, J.M. (1999) *Addressing interoperability in open hypermedia: the design of the Open Hypermedia Protocol*. New Review of Hypermedia and Multimedia 5(1), pp. 207-248.

# OHP architecture



# OHP data model



# Location specifiers (LocSpec)

Used to identify the position of an anchor within an object

- Byte offset within file
- Occurrence of a given string (i.e. generic link)
- Named location within content (c.f. HTML `id` attribute)

Exists within links (as part of the endpoint) and within nodes

Generalisation of Dexter's anchor values

- Mechanism for identifying locations in other formats
- Function to evaluate to identify anchor position

Grønbæk, K. and Trigg, R. (1996) *Toward a Dexter-based model for open hypermedia: unifying embedded references and link objects*. In HYPERTEXT '96: Proceedings of the the seventh ACM conference on Hypertext, pp.149-160.

# Evaluation

## Advantages

- Used in several demos and prototypes
- Commonly-accepted data model

## Disadvantages

- Shim architecture was naïve (expensive to implement)
- High message overhead

Anderson, K.M. et al (1997) *A critique of the Open Hypermedia Protocol*. Journal of Inline Digital Information, 1(2). | Available at: https://journals.tdl.org/jodi/index.php/jodi/article/view/5

# Link Integrity

http://www.flickr.com/photos/andreaswinterer/2460423132/

# Link Integrity

The endpoint of a link (source or destination) needs to define:

- a node
- (optionally) a position within a node (locspec)

If an endpoint fails to resolve to the place intended by the author, then it (and the link) is broken

Two types of link integrity failure:

- Dangling link
- Content reference

# The Dangling Link Problem

Occurs when an endpoint refers to an invalid node

Commonly the result of a node being moved or deleted

Possible mitigations:

- Responsibility of link creators (fine them if they don't keep them up to date?)
- Don't allows links to things that move
- Forward references
- Guaranteed names (PURL servers, DMS)
- The Hyper-G approach
- Link integrity checking agent (Spider)

# The Content Reference Problem

Occurs when an endpoint refers to a valid node, but to an incorrect location within that node

Commonly the result of changes/updates to the content of a node that are not reflected in links to that node

Possible mitigations:

- The publishing model – editing creates new resources
- Manual link editor
- Link service-aware editing tools
- Just-in-time link repairs
- Express specific link positions using queries
- Avoid specific links/anchors
- Versioning
- Use of diff files

# Responsible link owner or responsible system?

- Don’t bother: the Californian approach – it’s a social issue.
- Avoid the problem: use declarative link definitions
- Loosely coupled: give the author tools to sort the problem if they want
- Automated link repairs: just-in-time
- Tightly coupled: the Germanic approach – don’t let users have this freedom

# Evaluating open hypermedia

**Advantages**

- Applications not responsible for maintaining "foreign" markup
- Tailor linkbases to user needs (contexts)
- Generic links, etc
- Necessary for linking read-only media (e.g. CD-ROM, no permission)

**Disadvantages**

- Keeping links separately introduces potential consistency issues
- Integration with existing applications can be difficult

# Next Lecture: Hypertext Futures