BeComm Corporation

Technical Presentation

Why Strings?

Communications systems need to be:

Dynamic: networks, media types and end points are unpredictable

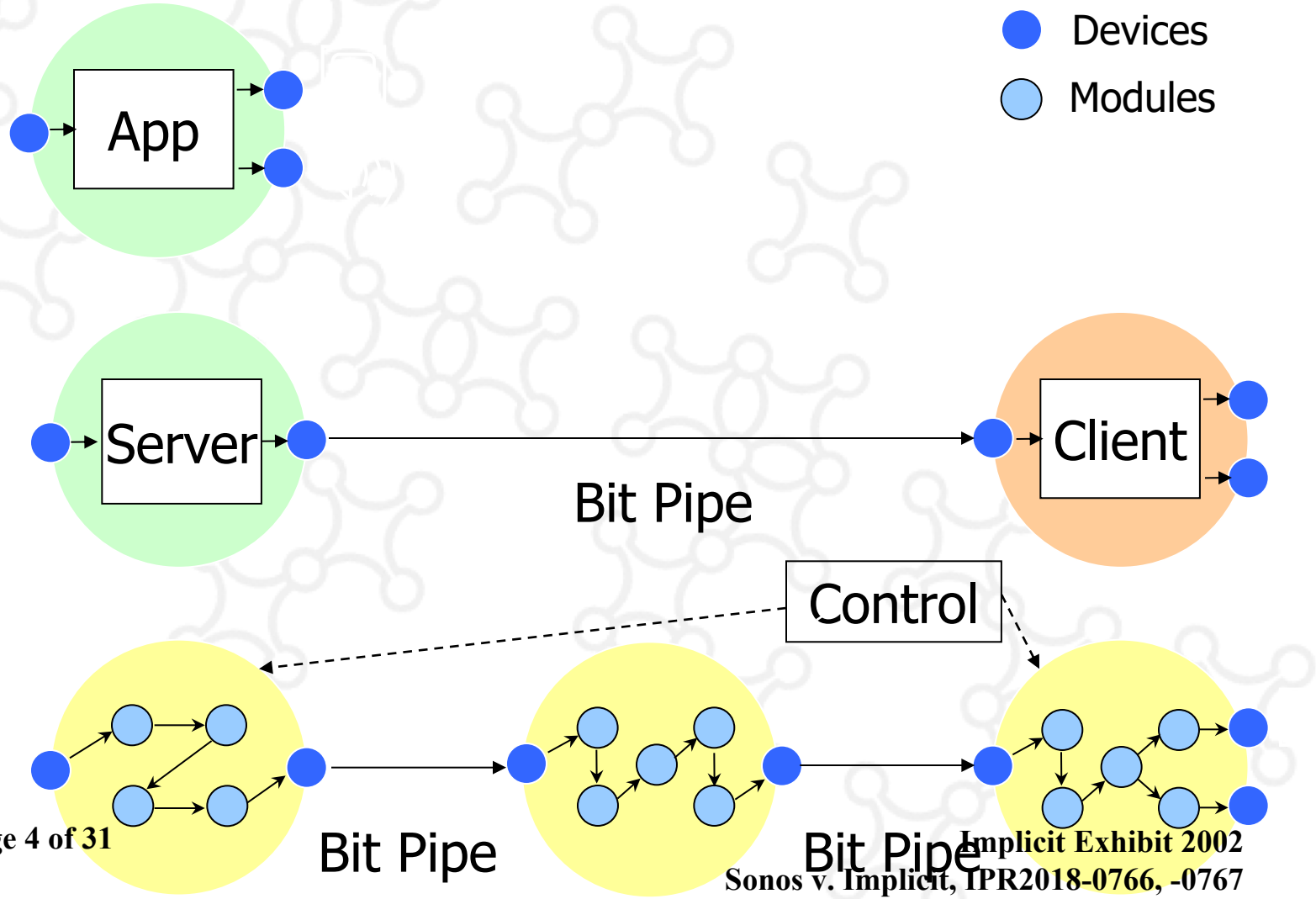
Distributed: content, resources and processing need to be distributed to maximum networked elements

Efficient: managing communications must be efficient for everything from overloaded back-end systems to resource constrained clients

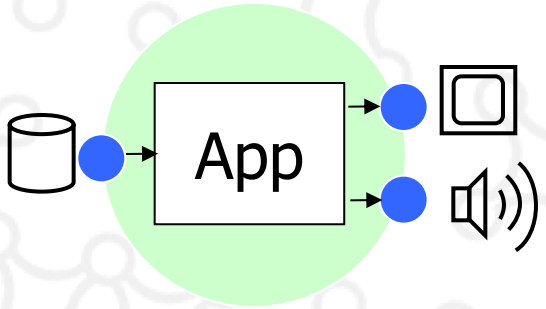
Potential Strings Enabled Systems

- Multimedia applications:
  - Video Players, Browsers, Audio Players, Video Conferencing, VoIP systems, Video on Demand, Digital/Personal VCR
- Networking:
  - NAT, Firewalls, Soft Routers, Soft Bridges, Remote Control
- Server:
  - File Servers, Media Servers
- Gateways:
  - Wireless Application Gateways, Media Gateways, Telephony Gateways, Web Portals, Content Management Systems
- Peer-to-Peer:
  - File Sharing systems, Collaboration systems, Distributed Processing systems

Application Evolution



A Traditional Video Player

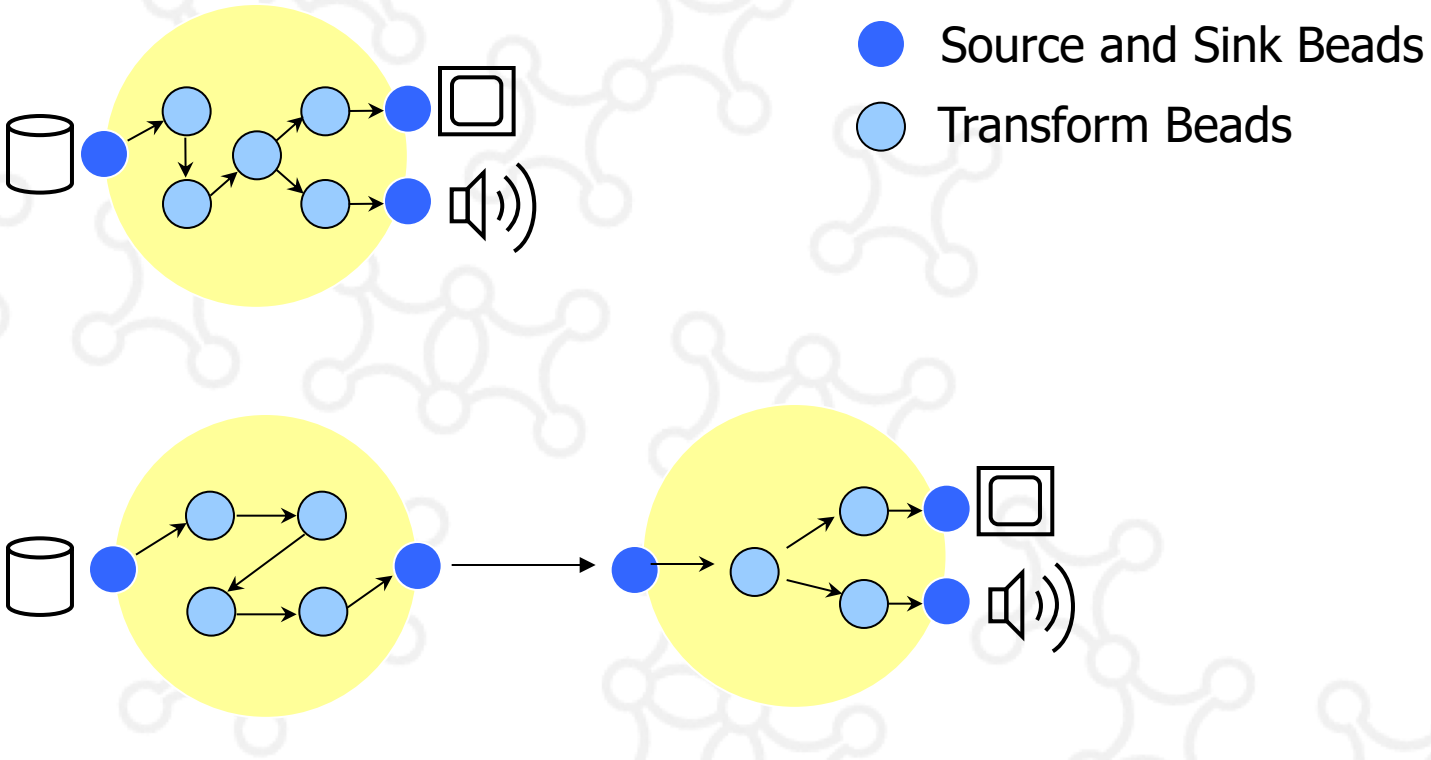


1. Application has built in knowledge of devices
2. Application features are tied together
3. Application components are not reusable in other contexts at runtime
4. Application cannot be decomposed and distributed

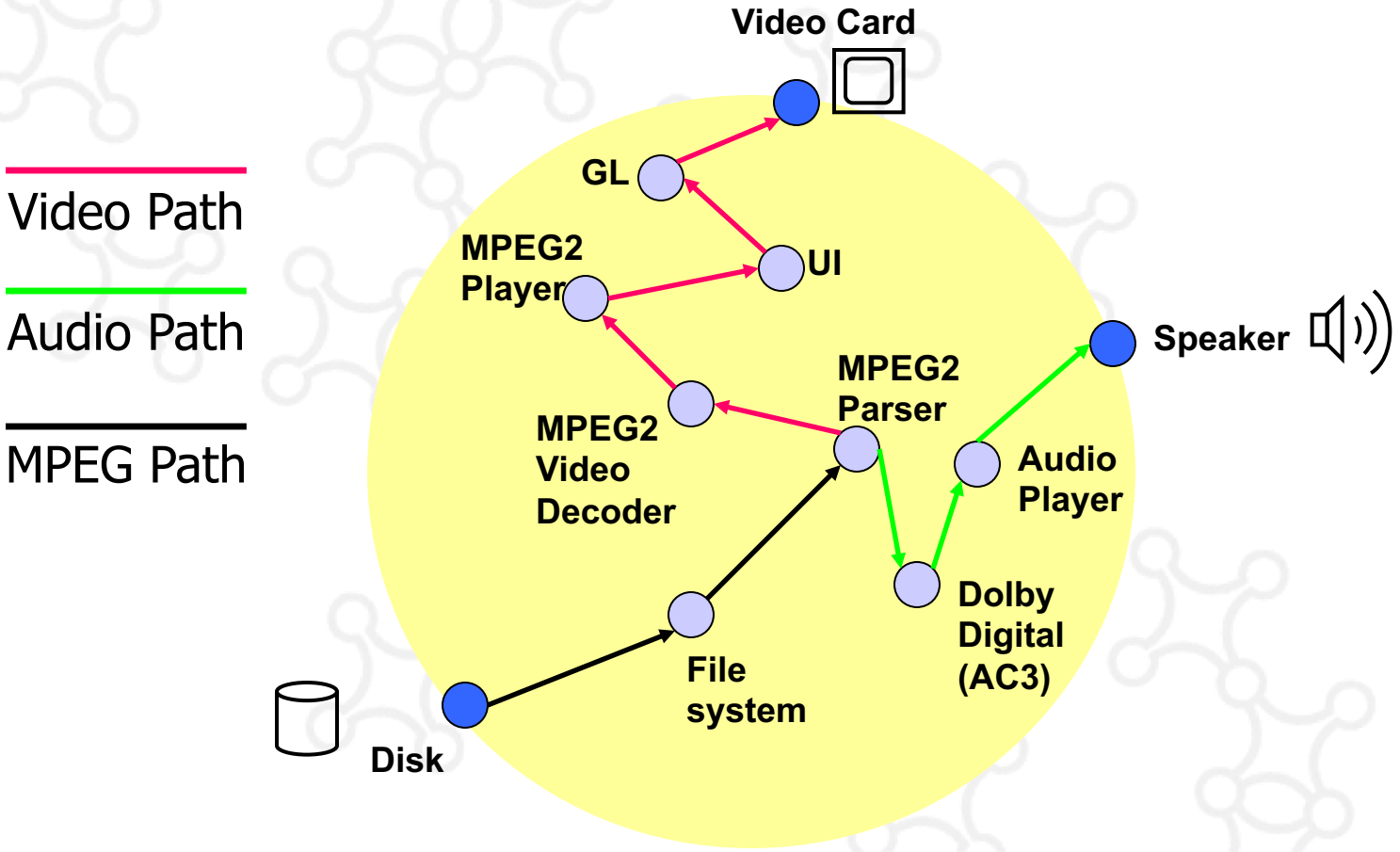
Strings Vernacular

- Beads
  - separate the software's internal activity from external relationship
- Labels
  - externalize description of Beads
- Mapping
  - Glue together Beads using their Labels
- Paths
  - global context of a set of mapped Beads

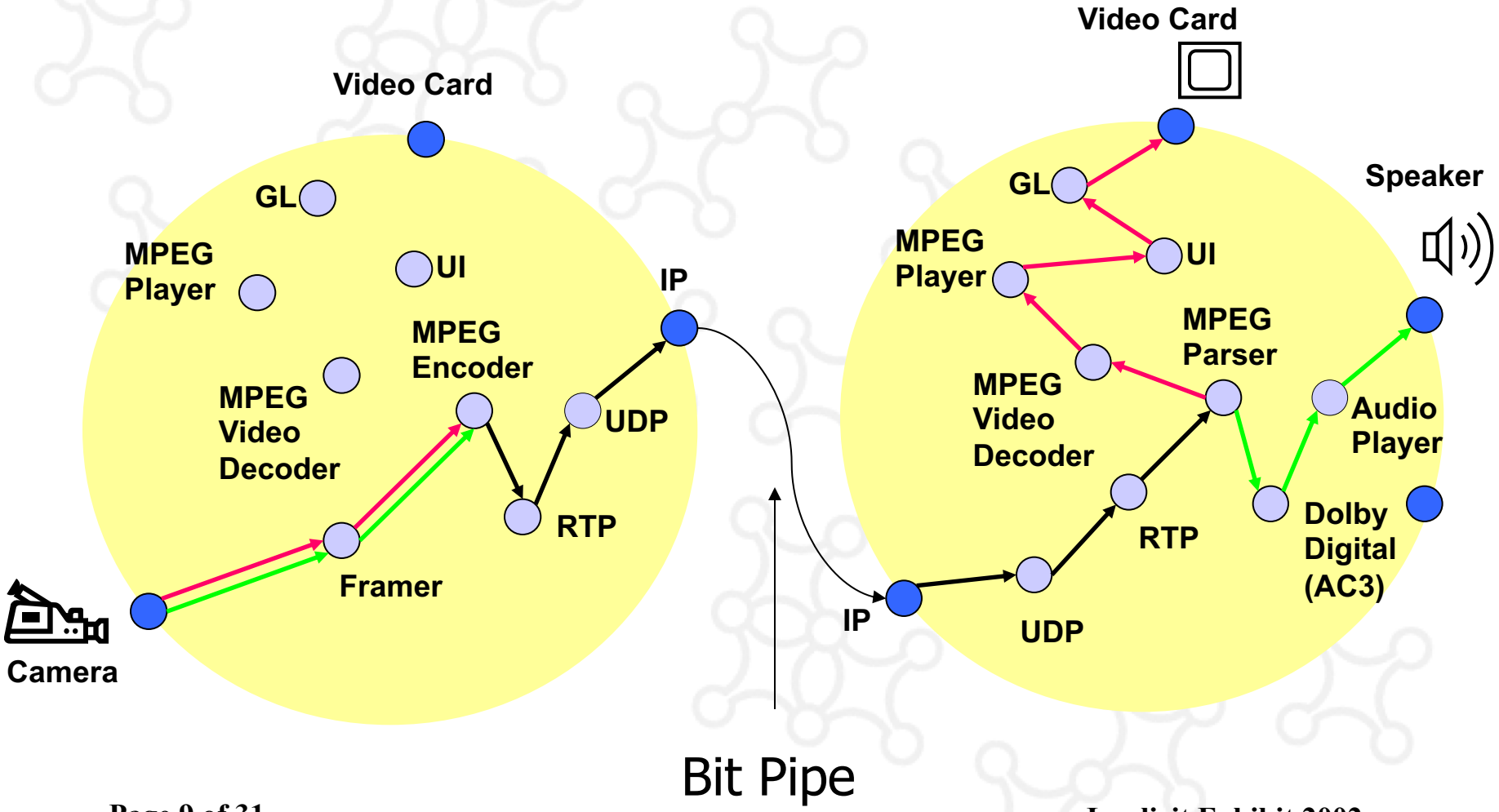
A Strings Video Player



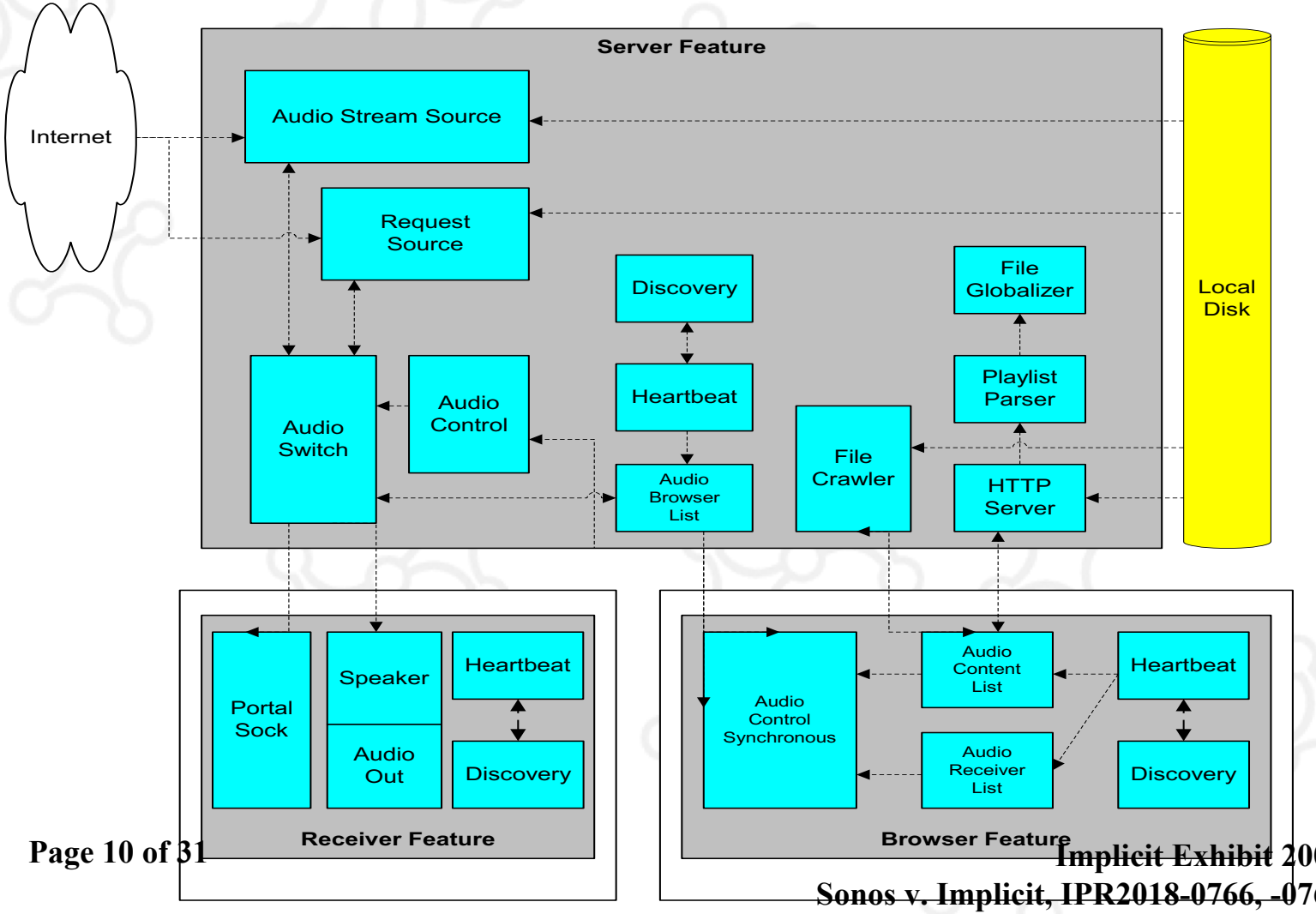
A Look Inside



A Distributed Video Player



Example: Distributed Media System



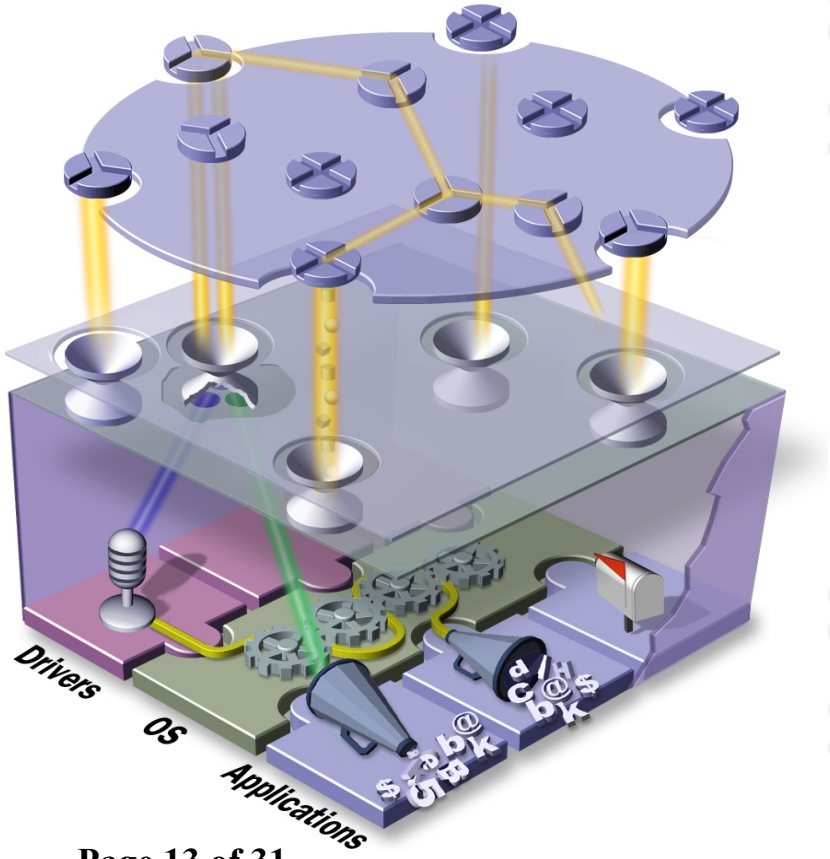
Benefits Over Traditional Systems

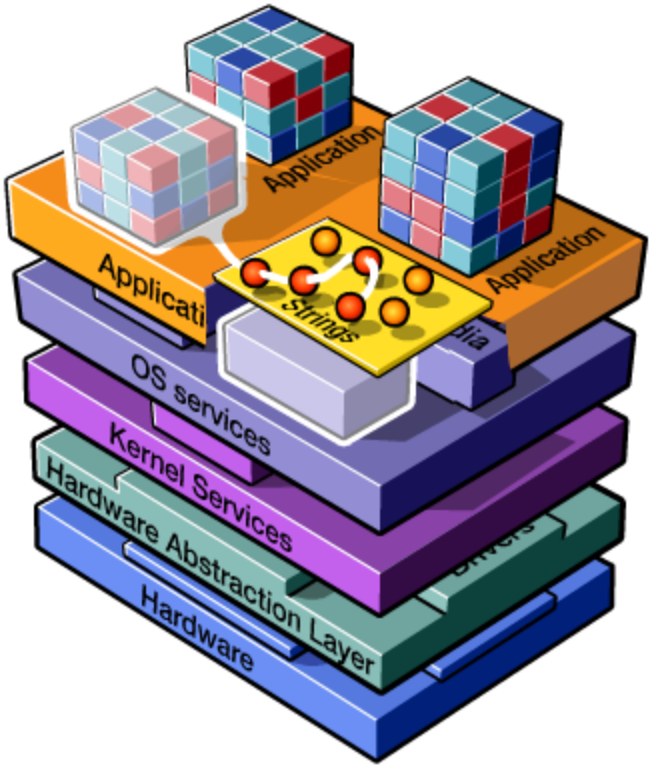
1. Device knowledge encapsulated in Beads
2. Features are re-configurable at initialization or run time
3. Network services are dynamically discovered
4. Beads can be re-organized to create a distributed service
5. Behavior can be modified without requiring new code

Platform Independence

- Interface Libraries provide access to external resources such as drivers, OS services and API
- Operating System Abstraction Layer (OAL) provides abstraction from underlying Kernel services such as access to physical memory, CPU scheduling and locks
- Strings manages scheduling, memory, timers and module loading for consistency across underlying platforms

Achieving Platform Independence





Beads

- Software components that encapsulate algorithms.
- Composed of a series of inputs and outputs
- Beads provide a service that when described to the core system can be used to create higher level services (features) on the fly.
- Focused only on encapsulating internal algorithms not external relationships.

Labels

- Labels are names used to tag information with query-able attributes.
- Labels are used to define inputs and outputs to Beads.
- Labels can be arbitrarily rich.
- A registry in Strings is used to name space created by the available labels.

Mapping

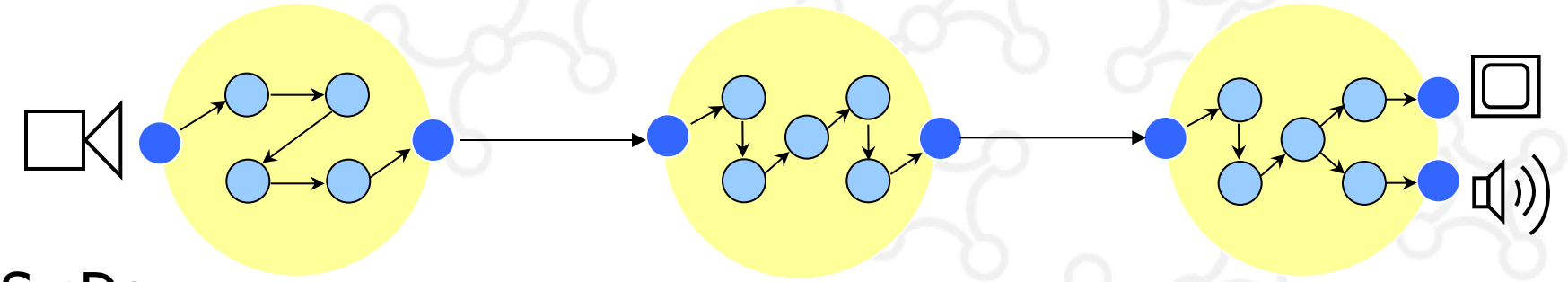
- The *Mapping Engine* is a system level service for determining what Beads are necessary to handle a given type of media within a given set of constraints.
- *The Map* is a set of rules to impose upon the graph search the Mapping Engine performs.
- Mapping is adaptive to the changing environment.
- Mapping is rules based meaning the determination of Beads can be influenced by external rules (security, network bandwidth etc..)

Paths

- Paths capture the global context of stream of data through a set of Beads.
- Paths can be extended across machines boundaries.
- Paths are efficient abstractions for managing streams of data.
- Paths are dynamically constructed and destructed.

Mapping

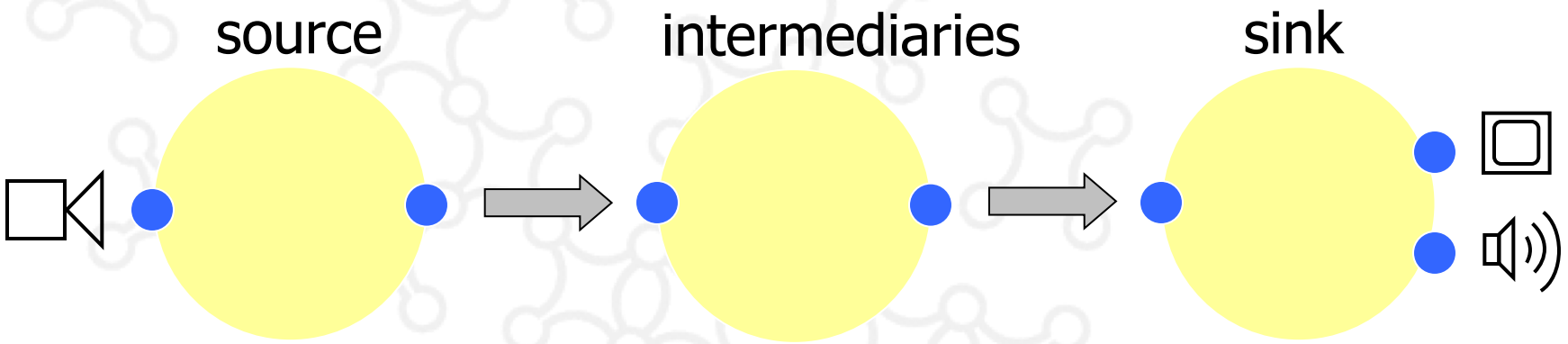
- Framework to string modules from source device to sink device(s)
- Necessary pieces
  - Resource Discovery and Route Selection
  - Code Module Installation



Input to the process

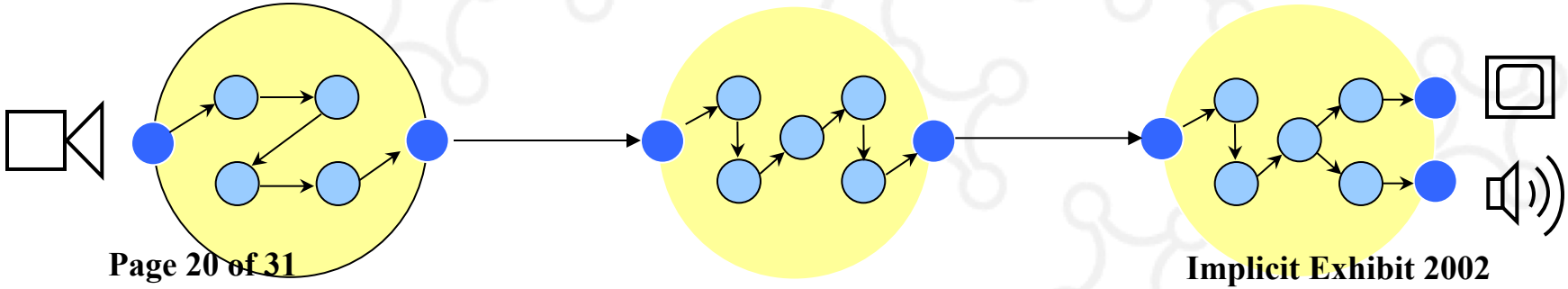
- Media object
  - Requirements to play
- User
  - Preferences
- Node
  - Capabilities and Devices (Resources)
- Systems Integrator
  - Rules of Composition

End-to-End Mapping



Resource Discovery and Route Selection

- Local Path Construction



Global Path Construction

- Resource Discovery
  - Directory Service (Node Capabilities)
- Route Selection
  - Functionality

    Locate sufficient resources for function chains

    Pattern Match (Rules with Node Capabilities)
  - Connectivity

    Issue: Limiting the search space

    Shortest Path/ K-shortest Paths / Overlay

Dynamic

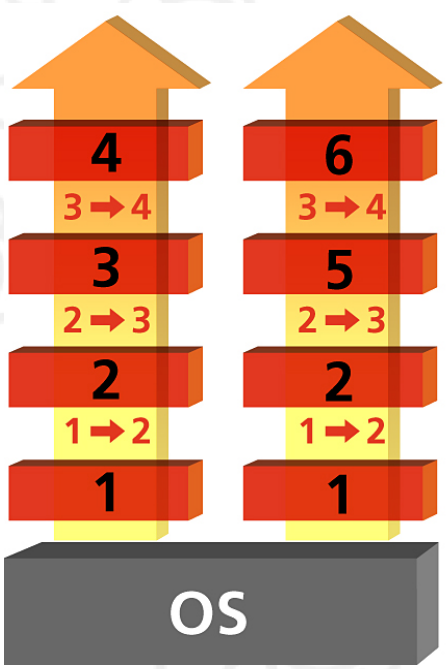
- Identify optimal set of Beads to handle a data flow
- Discover Beads on remote nodes to distribute processing
- Adjust Paths as necessary to adapt to changing QoS and resources
- Reroute Paths based on user input or system input

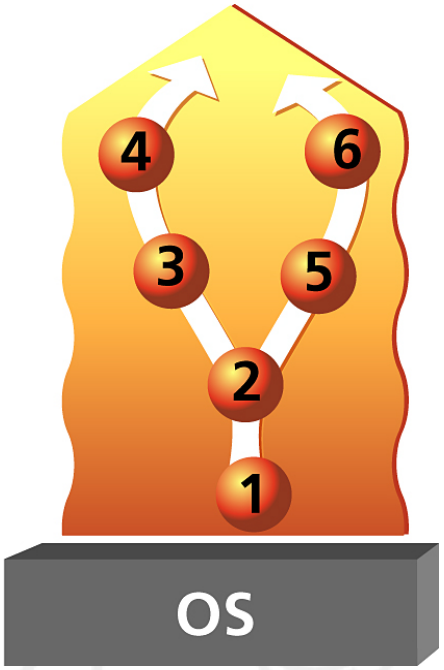
Distributed

- Name Space
- Distributing Content
- Distributing Resources
- Distributing Control
- End to End Paths

Efficient

- Vertical Scheduling
- QoS
- Logical Paths
- Physical Paths
- Code Reuse





Beads

- Beads are autonomous units of functionality which provide a well-defined service.
- The service provided by a Bead is described by its Schema.
- Instantiating a Bead requires configuring it to participate in higher-level services called Features.

Features

- A Feature is a collection of Beads and/or other Features
- Features are implemented in terms of the services provided by the elements (Beads & sub-Features) of the Feature.
- The service provided by a Feature can be described by the same Schema as the Schema used for a Bead
- Feature can be thought of as a composite Bead.

Service

- Features can be configured in such a way as to exported to the network
- Other hosts in the network can leverage that Feature either at runtime, via remote mapping
- …or at configuration time in the construction of higher level Features.

Example Schemas

```
<BEAD_SCHEMA name="mpegDecoder"
   sessionKeyType="device"
   configurationFileType="speakerConfigurationFileFor
   mat" />
   <INPUT_EDGE name="multimedia" format="mp4" />
   <OUTPUT_EDGE name="audio" format="pcm" />
   <OUTPUT_EDGE name="video" format="bmp" />
</BEAD_SCHEMA>
```

Strings Registry

- Rich & Diverse name space (e.g. files, devices, resources, media types, network addresses)
- Rich Querying language (e.g. where are my jazz files? What network service provides mpeg2 decoding with support for RTP framing?)
- Distributed Discovery (e.g. can support discovery services such as UPnP or Strings Discovery Protocol)
- Extensible (e.g. support NTFS, LDAP, Gnutella, UDDI)

Microsoft

- DirectX – multimedia framework
- NDIS – network framework
- COM – OO framework
- C#, .NET – distributed services framework
- UPnP – discovery services

Sun

- Java – OO framework for platform independent applications
- Jini – discovery services and API for distributed applications on Java
- JXTA – A peer to peer framework