Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

ANSAwise - Multi-media in Distributed Systems

Chris Mayers

Abstract

Designing and implementing multimedia applications in a distributed environment poses a combination of challenges - challenges which can be met in the near future.

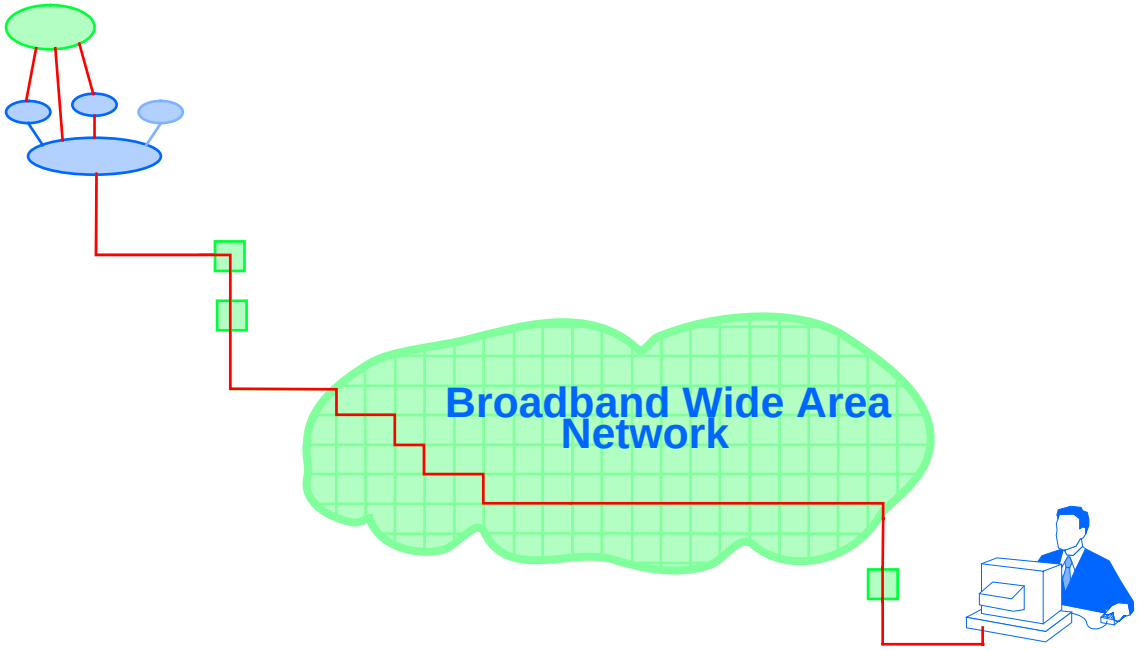
This module of the ANSAwise training programme shows how the specific needs of multimedia application translates into the computational and engineering requirements for streams, explicit binding, real-time support, QoS negotiation, and orchestration. It also shows how work in related areas (high-performance, real-time, QoS) provides ready-made solutions, and shows a practical application of these solutions.

[This module does not cover synchronous programming techniques.]

APM.1354.03 **Approved** 2nd April 1996
Briefing Note

Distribution:
Supersedes:
Superseded by:

Multimedia in Distributed Systems



In this session

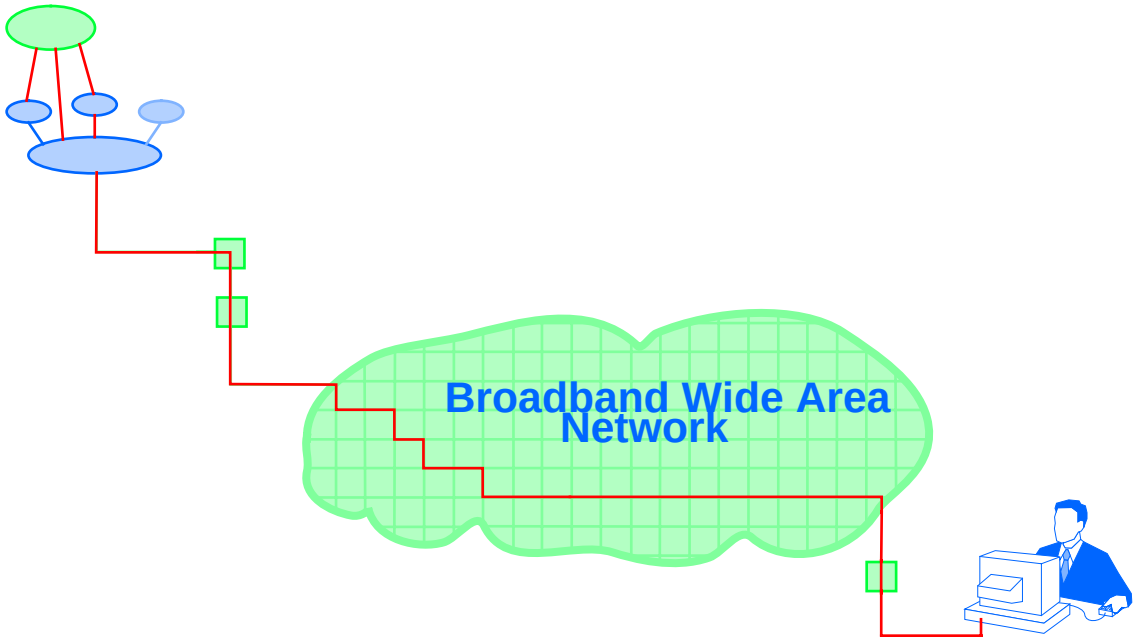
- Explain the needs of distributed multimedia systems
- Show how new concepts have evolved to accommodate these needs
- Explain the engineering mechanisms that support these concepts
- Show how multimedia depends on other facilities
  - real-time and replication support

Requirements for distributed multimedia

- **Support for continuous media**
  - **typically digital audio and video**
- **Specification and dynamic management of quality-of-service**
- **Group communications**
- **Real-time synchronization mechanisms**

Multimedia over a WAN

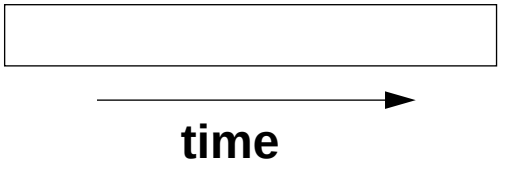
- **Guarantees must be end-to-end, and be implemented hop-by-hop...**



- **... possibly over a wide-area network**

Continuous media

- **Continuous media have a well-defined**
  - **begin**
  - **end**
  - **rate of delivery**



- **The begin and end correspond to the creation and deletion of a *binding***

A typical scenario - scientific collaboration

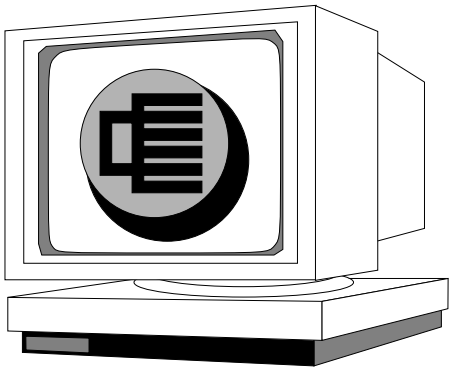
- Scientists discuss the input from remotely-sited industrial optical and electron microscopes
- Each scientist has an audio/video workstation
  - displaying slow-scan video from remote microscope devices...
  - ... quality of service varies
- The scientists can
  - record microscope output to disc
  - ...create multimedia documents (including voice annotations)
  - ... and send them as video mail

Implications of technical multimedia

- **Compared with video-conferencing, the modes of operation are more complex**
  - **instrument-to-person as well as person-to-person**
- **Quality of service requirements are more stringent**
  - **image degradation may not be acceptable**
- **Media may need to be recorded for future reference**

Replaying a recording

- When replaying in slow motion, what happens to the commentary?



  - the sound is slowed down?
  - the commentary is played starting at key points?
  - the commentary becomes more detailed?

Tackling the implications

- **These influence**
  - **the information model: labelling of continuous media with events, reflecting the logical structure of the content**
  - **the computational model: support for streams**
  - **the engineering model: support for explicit binding and synchronization**
- **And quality-of-service (QoS) support in all the models**

Synchronization

- **Stream synchronization between source and destination of a single stream is needed**
  - **for rate control and sequencing**
- **Lip-sync for audio and video is perhaps the most obvious**
  - **continuous synchronization...**
  - **... even though sources and destinations may be geographically separate**
- **Event synchronization, too**
  - **for example, to display a text caption...**
  - **... this action must take place in a *timely* manner**
- **Synchronization must be monitored to prevent and regulate drift**

Quality of Service for Multimedia

- **Stream synchronization requires support for a fundamental set of characteristics, including**
  - **throughput**
  - **delay**
  - **jitter**
- **Audio requires low throughput, low jitter**
- **Video requires high throughput, but is more tolerant of jitter**

QoS Parameters

- **Traffic characterization for variable bit rate (VBR) video**

Parameter	Description	VBR Reqt.
Peak rate arrival of cells	Maximum resources required by application at peak load	50 Mbit/sec
Average cell arrival rate	Average resources over connection duration	25 Mbit/sec
Burstiness	Peak cell rate/Average cell rate	2
Peak duration	Average duration of maximum load	10 msec

Recovery and Quality of Service Renegotiation

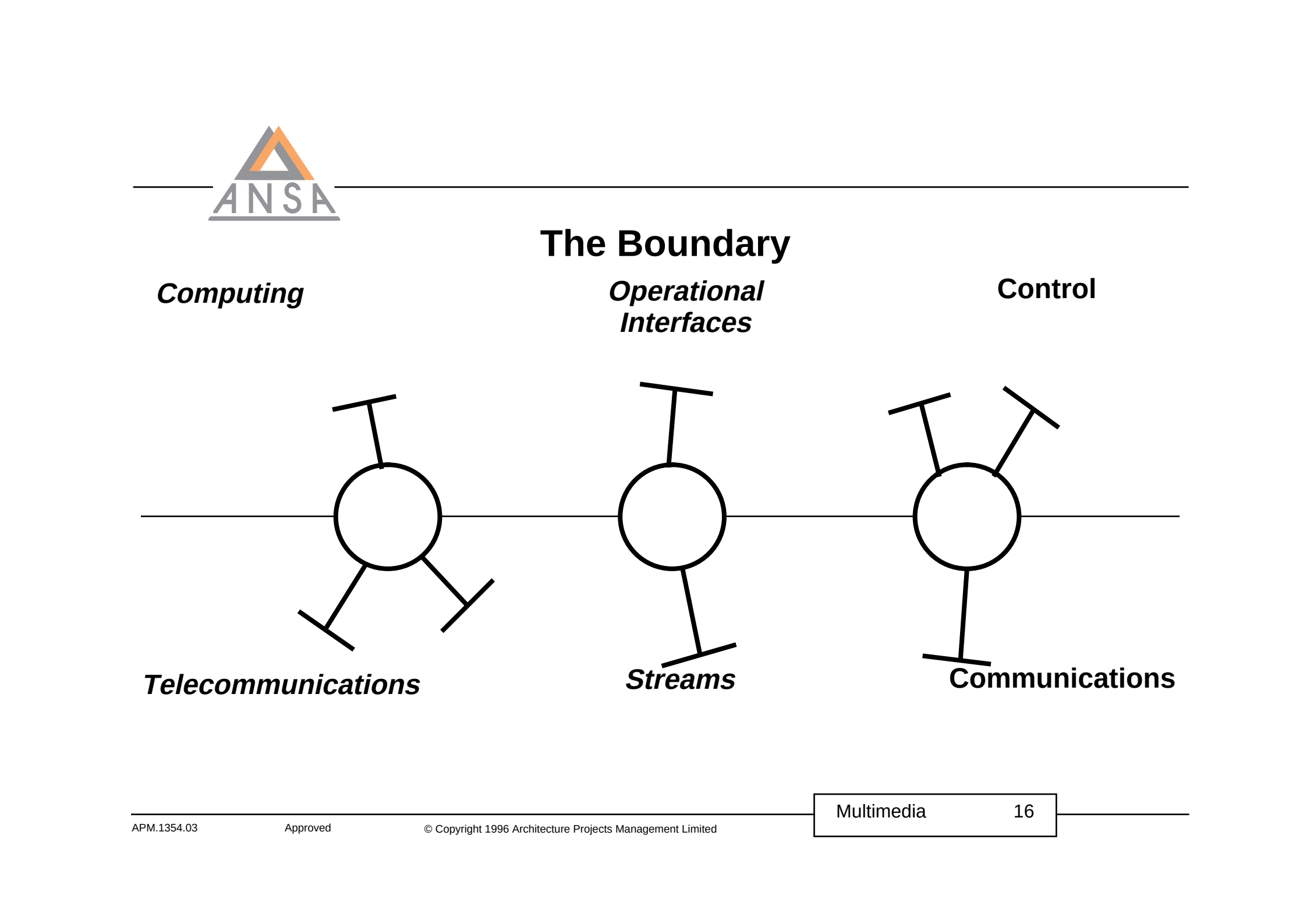
- **If the required QoS cannot be met, the application must intervene, for example**
  - **for video: degrade to grey-scale, or a still image**
  - **for audio: abandon the connection, and set up a new one**
- **But synchronization must be maintained**
  - **by *orchestration* mechanisms**

Orchestration mechanisms

- **These support**
  - **starting and stopping flows *precisely* together**
  - **creating related connections with compatible QoS**
  - **monitoring and regulation of related connections**

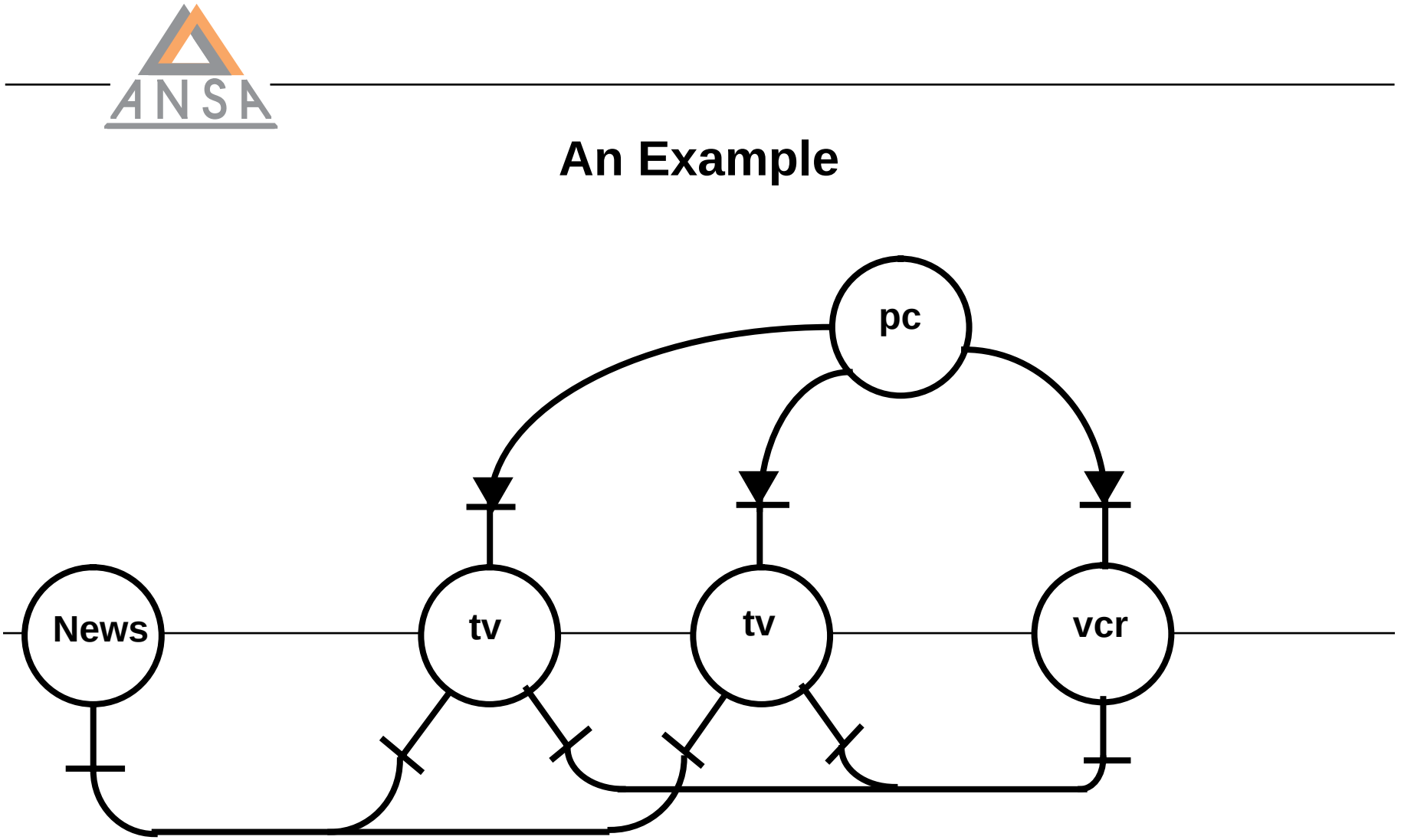
Objects in Distributed Multimedia

- Objects fit in naturally, to represent
  - streams (the continuous media)
  - their sources
  - their destinations
- Objects have types
  - preventing many configuration errors...
  - ...particularly important for complex stream configurations
- Objects exploit existing infrastructure
  - for example, security
  - ...and the transparency mechanisms (for example, for migration)
- Objects also control devices
  - for example, camera tilt and pan, or a display video window



Boundary Objects

- May have both operational interfaces and streams
- May have any number of these interfaces
- Such operational interfaces will typically have operations like:
  - start.flow ()
  - stop.flow ()
  - monitor ()



An Example

Defining operational interfaces

- CORBA IDL defines interfaces in terms of:
  - operations provided
  - ...their argument types
  - ...their result types
- Operational interfaces have an implied directionality
  - client performs invocation on server
  - arguments pass from client to server
  - results pass from server to client

The structure of streams

- Streams can be unidirectional or bidirectional
- Streams consist of a “bundle” of multiple flows
  - flows are unidirectional
- Flows consist of frames
  - the order of frames in a flow is preserved

Flows and Frames

- **Frames can be of any format**
  - **they can be like ‘video frames’ or ‘data frames’ (packets)...**
  - **...supporting both multimedia (audio/video) and control (sensor) formats**
- **Flows can contain multiple frame formats, supporting**
  - **frame-by-frame differencing (base+delta coding)**
  - **changes of compression algorithm or parameters between frames**
  - **application control of multiplexing and demultiplexing**
  - **in-band control**

Application Use of Streams

- An application may only need to *control* streams
  - for example, to direct a vcr feed to a tv...
  - ... the endpoint of the stream is a hardware device
- An application may wish to *process* the content of the stream
  - for example, to filter or reformat it
  - ...the endpoint of the stream is a software application

Application-defined Framing

- Some flows have natural formats
  - others do not
- There is a engineering decision on the best payload for each processing step
  - to handle different hardware framing formats
  - to handle different application formats for the same hardware formats
- This is simply a kind of “marshalling”
  - it can be handled by special “flow stubs”

A Proposed CORBA Mapping for Streams

- **Needs minor extensions to CORBA IDL**
  - **Streams are a slightly different kind of interface**
- **Frames could be defined as one-way operations**
  - **with only *in* arguments**
  - **... no result**
  - **... no user-defined exceptions**
- **The direction of flows could be shown as *in* or *out***

Establishing stream connections

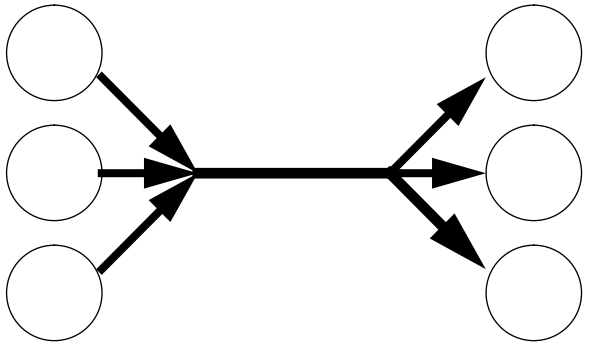
- **Establishing a stream connection requires explicit binding**
  - **it does not happen automatically**
- **This requires cooperation between all the parties involved**
  - **there may be multiple sources and multiple destinations for the flows**

Steps in explicit binding

- Every application has a local object called a binding manager
- The binding manager has an interface for creating endpoint binders
  - endpoint binders are also local objects

An example

- **Suppose we wish to build the following stream...**



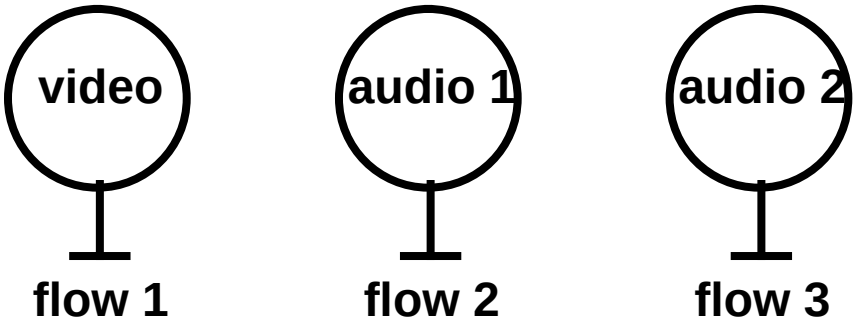
- **... two parties involved here**

Binding - the first step

- **All the parties involved need to ‘get together’**
  - **each party’s binding manager communicates with the others**
  - **the flows are checked to be compatible**
  - **... and any QoS specifications are checked for compatibility too**

Binding - the source end

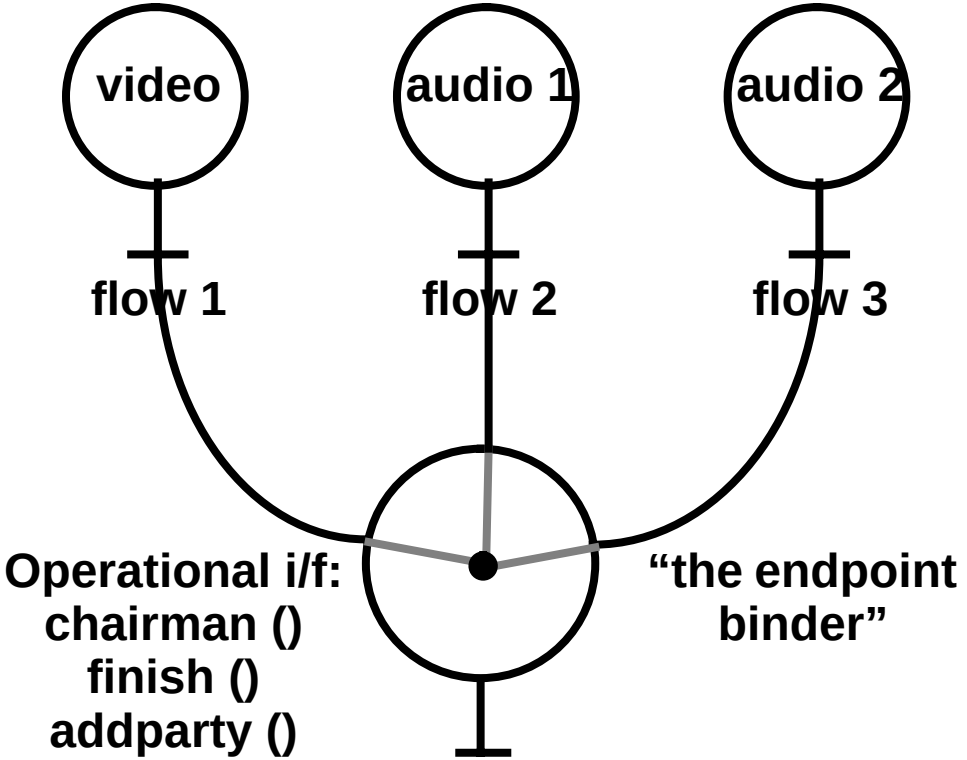
- At the source end, we have



- Pass the (local) binding manager the flows
- Binding manager creates a new object - “the endpoint binder”
  - which ties the flows together

The Endpoint Binder

- **The result looks like this:**



Binders Are Not Switches

- Binders are not like cross-connect switches or bridges
- Binders connect endpoints together
  - a network path
  - ...ultimate sources to ultimate destinations
- Switches connect ports together
  - a local link
  - ...local inputs to local outputs
- Binders need resources - for example, buffering
- Switches just tie up channels from inputs to outputs

Binding operations

- A binder may take multiple streams as input
- It may create one or more control interfaces
- The control interface may usefully have an “unbind” operation

Implicit and Explicit Binding

- **Operational Interface references contain sufficient information for binding to take place**
- **Binding of client and server can thus be implicit**
  - **it happens automatically...**
  - **...clients may be allowed some control over when this takes place**
- **However, streams are always explicitly bound**

Binding must be under application control

- **Binding is complicated**
  - **there are many useful configurations; QoS negotiation is also needed**
  - **... a small set of solutions would be inadequate**
  - **... it is too complex to build into the network**
- **But if the ORB did it, this would breach encapsulation**
  - **it would defeat security...**
  - **... if an object cannot control what it connects to, how could it secure itself?**

Binding Templates - simplifying application control

- Application programmers will use pre-configured bindings
  - based on generic *binding templates*
- One CORBA proposal is to extend the ORB with “stream object adapters”
  - different adapters for different technologies
  - ... HTTP, in-process FIFO streams,...
  - different adapters for different policies
  - ... synchronization, security, management,...
- These can exploit CORBA inheritance and language features
  - via C++ templates in the language mapping

Real-time support

- In practice, the flow of continuous media can work reasonably well even distributed over non-deterministic LANs
  - Ethernet can support 6 voice channels (3 users), or slow-scan video
- Control of continuous media can be problematic
  - starting a video play requires a large number of RPC invocations...
  - ... which must not be lost
- However, we already have the solution to this
  - real-time support for bounded delay with Timed RPC and resource reservation

Summary

- **Supporting multimedia requires extensions to the computational model and to the engineering model**
  - **Streams, explicit binding, and QoS**
- **These extensions are quite feasible...**
- **...but high-level support would help the application programmer**
  - ***templates* for configuring streams and bindings**
- **General-purpose workstation hardware is sufficient - up to a point**
- **Multimedia interoperability will require new standardization**

More information?

- **For more information**
  - **on multimedia, see *Integrating Multimedia into the ANSA Architecture* (TR.028)**
  - **on binding, see *The ANSA Binding Model* (APM.1314)**
  - **on a proposed CORBA extension for streams, see *CORBA and Streams: a White Paper* (APM.1684)**