Using Quality Objects (QuO) Middleware for QoS Control of Video Streams

BBN Technologies
Cambridge, MA
http://www.dist-systems.bbn.com/tech/QuO/

Craig Rodrigues
crodrigu@bbn.com

OMG’s Third Workshop on Real-Time and Embedded Distributed Object Computing
January 7-10, 2002
Burlingame, California, USA
US Navy UAV Characteristics

UAV Basic Scenario

- Video feed from off-board source (unmanned aerial vehicle)
- Video Distributor sends video to hosts via ship’s network
- Users’ hosts receive video and display it
- Users interact with UAV in real time
Example Dynamic Variations in UAV Mission Requirements

Different mission requirements require optimizing an aspect or aspects, possibly trading off others.

<table>
<thead>
<tr>
<th><strong>Timeliness</strong></th>
<th>End-to-end latency + processing time at distributor and display, must be below threshold</th>
<th>Timeliness Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>• An out-of-the-window view of UAV imagery (e.g., piloting, targeting)</td>
<td></td>
<td>L(U⇒D) + P(D) + L(D⇒R) + P(R) &lt; T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Importance</strong></th>
<th>Highest priority frames must reach the receiver</th>
<th>Importance Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Receiving or collecting sufficient imagery (e.g., surveillance)</td>
<td></td>
<td>{ f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Fidelity</strong></th>
<th>Image precision at display must approximate that at the UAV source</th>
<th>Fidelity Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Determining relative importance of time-critical targets (e.g., commander)</td>
<td></td>
<td>Fidelity(R) / Fidelity(U) &gt; Q</td>
</tr>
</tbody>
</table>

*Trading off these properties, while maintaining invariants, under dynamic conditions is crucial to mission success.*
Example Variations & Adaptations in UAV Scenario

**Mission requirements of UAV scenario**

**Timeliness**
- Maintain an out-of-the-window view of UAV imagery

**Importance**
- Important imagery must be received

**Fidelity**
- Image fidelity must be maintained

**NETWORK RESERVATION**
- Condition: excessive Network load
- Action: Use IntServ to reserve bandwidth

**LOAD BALANCING**
- Condition: Excessive CPU load
- Action: Migrate distributor to a lightly loaded host

**DATA FILTERING**
- Condition: When excessive network or CPU load
- Action: Drop selective frames

**Dynamic Variations in Operating Conditions**

*NETWORK RESERVATION*
- Condition: excessive Network load
- Action: Use IntServ to reserve bandwidth

*LOAD BALANCING*
- Condition: Excessive CPU load
- Action: Migrate distributor to a lightly loaded host

*DATA FILTERING*
- Condition: When excessive network or CPU load
- Action: Drop selective frames
UAV Application:
A View of the Software Components

Common middleware services
- TAO CORBA A/V Service
- QuO
- AQoSA
- TAO Naming Service

Reused Off the Shelf Software
- DVDView video player
Functional description of UAV architecture

1. **Video Source Process**
   - **Functionality**
     - Read bytes from a file
     - Convert into frames
     - Send out pipe
     - Timing and sequencing of frames sent out

2. **Video Distributor Process**
   - **Base Functionality**
     - Receive frames
     - Send frames to registered receivers
   - **QuO Functionality**
     - **QuO Delegate:**
       - Frame filtering (i.e., frame dropping)
     - **QuO Sysconds and Contract:**
       - Measuring frame rate
       - Measuring resource usage and availability (CPU and network)
       - Adapting to performance degradation (e.g., frame dropping, load balancing)

3. **Display Proxy Process**
   - **Base Functionality**
     - Receive frames
     - Display frames on the screen
   - **QuO Functionality**
     - **QuO Delegate:**
       - Removing time stamp
       - Removing sequencing
       - Discarding frame if late or out of sequence
     - **QuO Sysconds and Contract:**
       - Measuring frame rate

4. **Video Display Process**
   - **Video Display Host 1**
   - **Video Display Host 2**
   - **Video Display Host N**
COTS Middleware Components used in the UAV Software

• **Quality Objects (QuO)**
  – provides higher level programming model for specifying application-level QoS

• **CORBA Audio/Video Service**
  – abstracts away network programming concerns such as connection management, can be used to specify network level QoS
Quality Objects (QuO)

• Provides a higher level programming model for specifying application QoS
  – operating regions specified in QuO Contracts
  – transitions between regions trigger adaptive behaviors

• Support for different middleware architectures
  – CORBA (Java and C++)
  – Java RMI
  – local method call
QuO Adds Specification, Measurement, and Adaptation to the Distributed Object Model
CORBA Audio/Video Streaming Service

• **Goals of OMG CORBA A/V Service Specification**
  – Define standard mechanisms for:
    Stream Establishment
    Stream Control
    Multiple Flows
    Multiple Protocols
    QoS

• **Goals of TAO A/V Streaming Service Project**
  – Implement OMG CORBA A/V Service Specification using TAO
UAV Application High Level Design

Video Source Process

UAV Video File

Video Source Process

Video Distributor

QuO

VIDEO DISTRIBUTION HOST

Video Stream

Video Display Proxy

Video Display

VIDEO DISPLAY HOST

Source Stream Endpoint (distributor)

Stream Adaptor

Sink Stream EndPoint (Display)

Stream Adaptor

QuO measurement and control

TAO A/V Streaming Service

ACE AQoSA API

RSVP-enabled routers

Video Stream

Video Stream

Reservation request

Reservation notification

flows

QoS updates

reservations

request/reservation

accept/reject

event notifications
UAV Application:
A View of the Software Components

- **Common middleware services**
  - TAO CORBA A/V Service
  - QuO
  - AQoSA
  - TAO Naming Service

- **Reused Off the Shelf Software**
  - DVDView video player
• QuO provides higher level feedback and control as video frames are transmitted from sender to receiver
RSVP fragmentation problem solved by use of QuO Qoskets

- MPEG in UDP packets were being fragmented by IP.
- Routers did not know that fragmented IP packets were part of a RSVP reservation of UDP packets.
- Solution: QuO delegates for fragmentation/reassembly, so UDP packets are not fragmented at IP level.
RT-CORBA and UAV application

- Distributor will need to perform real-time processing of video frames and/or sensor data which may accompany the video, or video frames
- Distributor invokes a method on an RT-CORBA servant which performs data processing
- Thread priority of task is requested in the CORBA invocation
What is Diffserv?

IP Datagram Header

<table>
<thead>
<tr>
<th>Version (4-bits)</th>
<th>Header Length (4-bits)</th>
<th>Diffserv Field (8-bits)</th>
<th>Total Length (16-bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification (16-bits)</td>
<td>Flags (3-bits)</td>
<td>Fragment offset (13-bits)</td>
<td></td>
</tr>
<tr>
<td>TTL (8-bits)</td>
<td>Protocol (8-bits)</td>
<td>Checksum (16-bits)</td>
<td></td>
</tr>
<tr>
<td>Source Address (32-bits)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address (32-bits)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Diffserv field**: 6 bits of Diffserv Codepoint, 2 bits ECN
- **Each DSCP (0-63) specifies a Per-Hop-Behavior (PHB)**, which is a type of router-level QoS (RFC 2475)
Specifying Diffserv QoS properties using the TAO AV Service

- CORBA IDL for AVStreams contains a method for changing the QoS characteristics of a stream:

```c++
typedef sequence<string> flowSpec;

struct QoS
{
    string QoSType
    CosPropertyService::Properties QoSParams;
};

typedef sequence<QoS> streamQoS;

interface Basic_StreamCtrl
{
    boolean modify_QoS(inout streamQoS, in flowSpec) raises (noSuchFlow, QoSRequestFailed)
}
```

- We have modified the TAO AV service, and use modify_QoS() to specify Diffserv codepoint values in video streams. AV service contributions are integrated in TAO 1.2.
- CORBA works well as a QoS signalling platform.
Creating RSVP and Diffserv Reusable Qoskets

- Desirable to create reusable network level QoS components ("Qoskets"), to make distributed, QoS enabled applications easier to develop.
- Code for RSVP qosket is currently being refactored out of existing application.
- Diffserv qosket needs to be rewritten.
- Perhaps a good use for the CORBA Component Model?
Future Directions

- Use more RT-CORBA features to deliver end-to-end QoS
- Use the CORBA AV service to send multiple simultaneous data streams (video, sensor readings)
- Create reusable Qosket components for RSVP, Diffserv, and other QoS properties
- CORBA seems to work well as a signalling platform for QoS. Can we use it to request network level QoS instead of protocols like RSVP?
- How can we integrate with Resource Managers for requesting end-to-end QoS: network, CPU resources,..?
Distribution information

- Software available from our web site:
  
  http://www.dist-systems.bbn.com/projects/AIRES/UAV/

- Help available from: quo-help@bbn.com

- Requires latest version of TAO ORB, version 1.2, available from:
  
  http://deuce.doc.wustl.edu/Download.html