QoS Control of Video Streams Using Quality Objects and the CORBA Audio/Video Service

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

OOMWorks
St. Louis, MO
http://www.oomworks.com
Yamuna Krishnamurthy
Irfan Pyarali

OMG’s Second Workshop on Real-Time and Embedded Distributed Object Computing
June 4-7, 2001
Herndon, Virginia, USA
Unmanned Air Vehicle

uav.navair.navy.mil/home.htm
Prototype Architecture

- Video feed from off-board source (UAV)
- Distributor sends video to hosts on ship’s network
- Users’ hosts receive video and display it
- Users send commands to UAV to control it
Why is Adaptation Necessary?

• Limited resources
  – CPU and network bandwidth are shared by many applications, not just video
  – Some applications are of higher priority than others, so it becomes necessary to adapt in response to scheduling and resource management decisions
UAV adaptive tactics

NETWORK RESERVATION
• Under excessive Network load - Use IntServ/DiffServ to reserve bandwidth

DATA FILTERING
• Excessive network or CPU load - Drop frames

LOAD BALANCING
• Excessive CPU load - Move distributor to more lightly loaded host
Why use a Framework?

- 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
- ACE QoS API (AQoSA)
  - uniform API for QoS (RSVP, GQoS)
UAV application design

Video Source Process

CORBA A/V Streaming Service

Video Distributor

Video Display Proxy

Video Display

UAV File

UAV SIMULATION HOST

VIDEO DISTRIBUTION HOST

Frame filtering commands

QuO Contract

Reservation requests

Observed throughput

VIDEO DISPLAY HOST 1

VIDEO DISPLAY HOST 2

VIDEO DISPLAY HOST N

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

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

Video Source Process
- Read bytes from a file
- Convert into frames
- Send out pipe

Functionality
- Base Functionality
  - Receive frames
  - Send frames to registered receivers
- Other intertwined aspects
  - Functional:
    - Frame filtering (i.e., frame dropping)
  - QoS:
    - Measuring frame rate
  - Real Time:
    - Time stamping
    - Sequencing

Display Proxy Process
- Base Functionality
  - Receive frames
  - Display frames on the screen
- Other intertwined aspects
  - QoS:
    - Measuring frame rate
  - Real Time:
    - Removing time stamp
    - Removing sequencing
    - Discarding frame if late or out of sequence

Video Display Process
- Base Functionality
  - Receive frames
  - Display frames on the screen
- Other intertwined aspects
  - QoS:
    - Measuring frame rate
  - Real Time:
    - Removing time stamp
    - Removing sequencing
    - Discarding frame if late or out of sequence

Video Distributor Process
- Functionality
  - Read bytes from a file
  - Convert into frames
  - Send out pipe
- Other intertwined aspects
  - Timing: periodic delivery of frames

VIDEO DISTRIBUTION HOST

VIDEO DISPLAY HOST 1

VIDEO DISPLAY HOST 2

VIDEO DISPLAY HOST N
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
Specifying QoS behaviors with QuO

C++ Application Code

```cpp
class Sender {
public:
    /// Constructor
    Sender(void);

    /// Method to pace and send data from a file.
    int pace_data(CORBA::Environment &);

    /// Accessor to the connection manager.
    Connection_Manager &connection_manager(void);

    /// Amount of debugging info to print out: 0 = none, 10 = lots
    int debug_level_;  

private:
    Connection_Manager &connection_manager(void);
    int parse_args(int argc, char **argv);
};
```

```
behavior Sequence () {
    remoteObj->send_frame(timed_msg); last_time_processed = tsp;
}
```

```
behavior Filter () {
    remoteObj->send_frame(seq_msg);  // insert sequence number
    seq_number++;  
}
```

```
Video Distributor Process
```

```
Video Source Process
```

```
UAV SIMULATION HOST
```

```
VIDEO DISTRIBUTION HOST
```

```
VIDEO DISPLAY HOST 1
```

```
VIDEO DISPLAY HOST 2
```

```
VIDEO DISPLAY HOST N
```

```
Display Proxy Process
```

```
Video Display Process
```
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
CORBA A/V Service Overview

Stream EndPoint
(Sink)

Multimedia Stream

POA

ORB

Stream Interface
Control Object

Control and
Management
Objects

Stream Interface
Control Object

Stream EndPoint
(Source)

Stream Adaptor

Stream Adaptor
CORBA A/V Service Overview

- **StreamCtrl**
  - controlling the stream

- **MMDevice**
  - interface for logical or physical device

- **StreamEndPoint**
  - network specific aspects of a stream

- **Vdev**
  - properties of the stream
QoS Enabled TAO A/V Service

- **TAO AV QoS Framework**
  - AQoSA
  - QoS Mapping
  - QoS Monitoring
  - QoS Adaptation

- **ACE QoS API (AQoSA)**
  - RAPI
  - GQoS
AQoSA API

- **Binds flows to reservations** in a uniform and portable way
- **Encapsulates** application’s notion of the underlying network **QoS**
- **Separates QoS properties** of its sessions from low-level socket data transfer aspects
- Allows the application to **specify** and **request** QoS. Be notified if the QoS was not available
- **Updates** itself based on QoS notifications
- **Portably** wraps the QoS parameters
- **Implementation** targets (RSVP) Resource reSerVation Protocol
  - RAPI & GQoS
AQoSA Architecture
Connection Establishment

bind_devs (aMMDevice, bMMDevice)

StreamCtrl

create_A
A_Endpoint, A_VDev
connect (with QoS)

create_B
B_Endpoint, B_VDev

QoS Monitor

 AQoSA

B_Endpoint

register session
and callback

qos events

configure

create
session

reserve

path

reserve

path
Change in QoS
Integrating TAO A/V in the UAV Application

- More flexible than ad-hoc socket code
- Facilitated additional features:
  - Multiple Viewers
  - Multiple receivers
  - Movable distributors
  - Network level QoS (with AQoSA)
Further References and Obtaining Software

- http://www.dist-systems.bbn.com/tech/QuO/
  - information about QuO framework
  - to obtain QuO, send e-mail to quo-help@bbn.com
- http://www.cs.wustl.edu/~schmidt/TAO.html
  - source code for TAO ORB, CORBA A/V service, and AQoSA
  - UAV application uses TAO 1.1.15 or higher (TAO 1.2 major release is imminent)
  - further information can be obtained via e-mail from av@oomworks.com
- Contact information:
  - Craig Rodrigues, crodrigu@bbn.com, 1-617-873-4725
  - Yamuna Krishnamurthy, yamuna@oomworks.com, 1-314-726-1368