

Building a (Node) Service

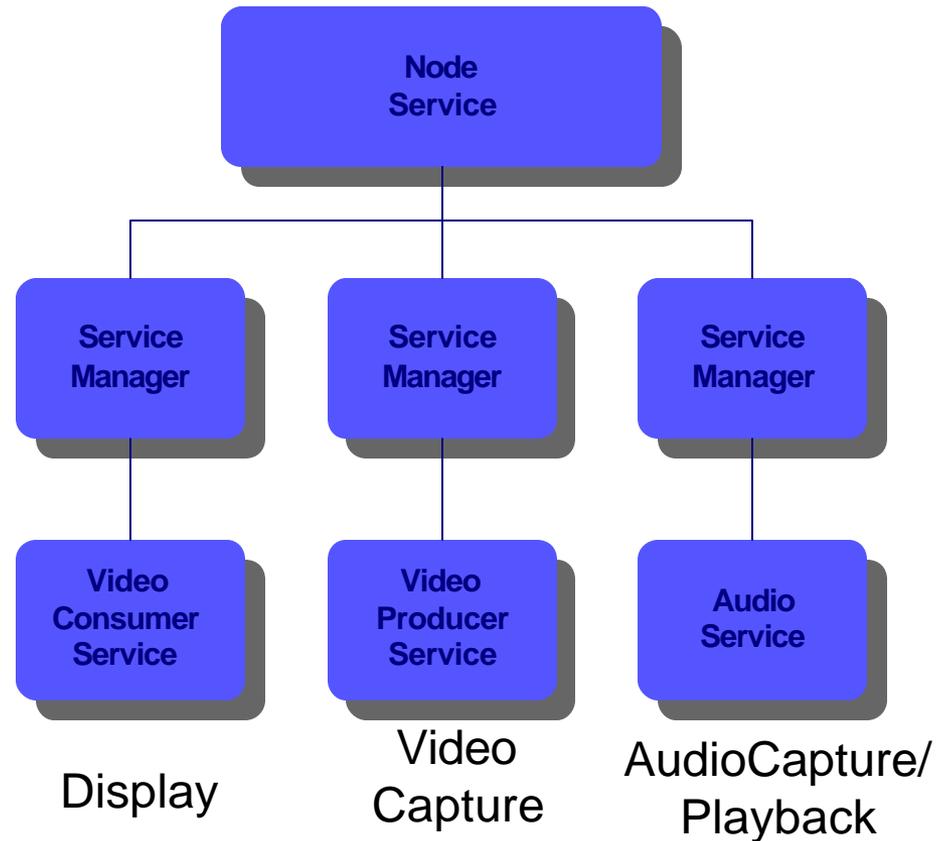
- Services Overview
- Review VideoProducerService
- Add Camera Control



Retreat 2003

Services Overview

A Service is executed on a machine in a Node to provide access to the machine's resources



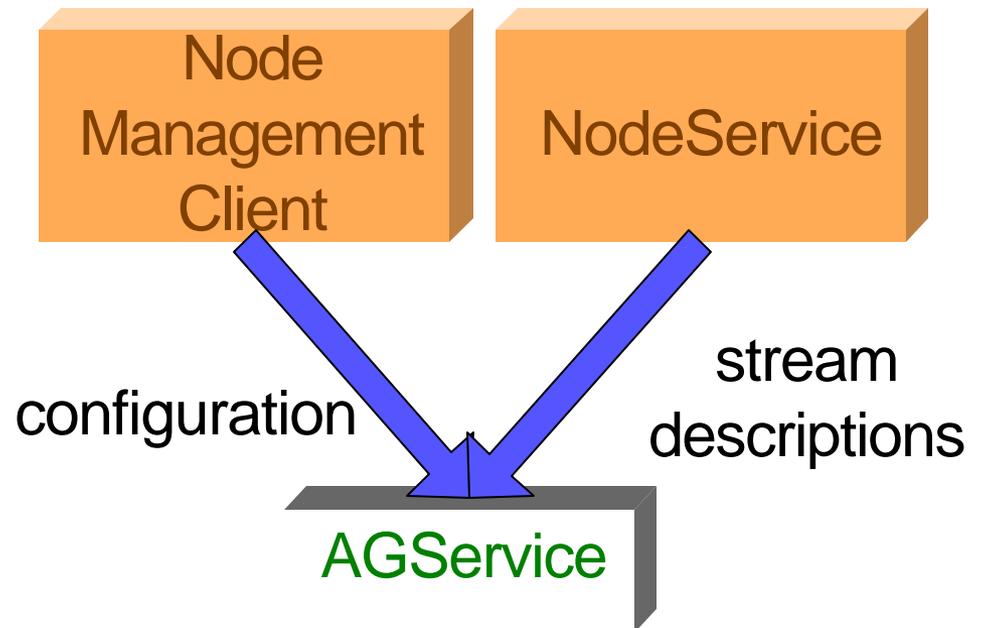
Service Responsibilities

- Adhere to a specific SOAP interface
- Describe capabilities
(e.g. video/h261/25fps,
audio/L16/16kHz)
- Handle stream description updates
(network addresses)
- Handle configuration updates



AGService Interoperation

- Node Management client depends on AGService interface for configuration
- NodeService depends on AGService interface to transmit stream descriptions



AGService Interface

Interface for Services to implement

- `def __init__(self)`
- `def Start(self)`
- `def Stop(self)`
- `def ConfigureStream(self, streamDescription)`
- `def SetAuthorizedUsers(self, authorizedUsers)`
- `def GetCapabilities(self)`
- `def GetResource(self)`
- `def SetResource(self, resource)`
- `def GetExecutable(self)`
- `def SetExecutable(self, executable)`
- `def SetConfiguration(self, configuration)`
- `def GetConfiguration(self)`
- `def IsStarted(self)`



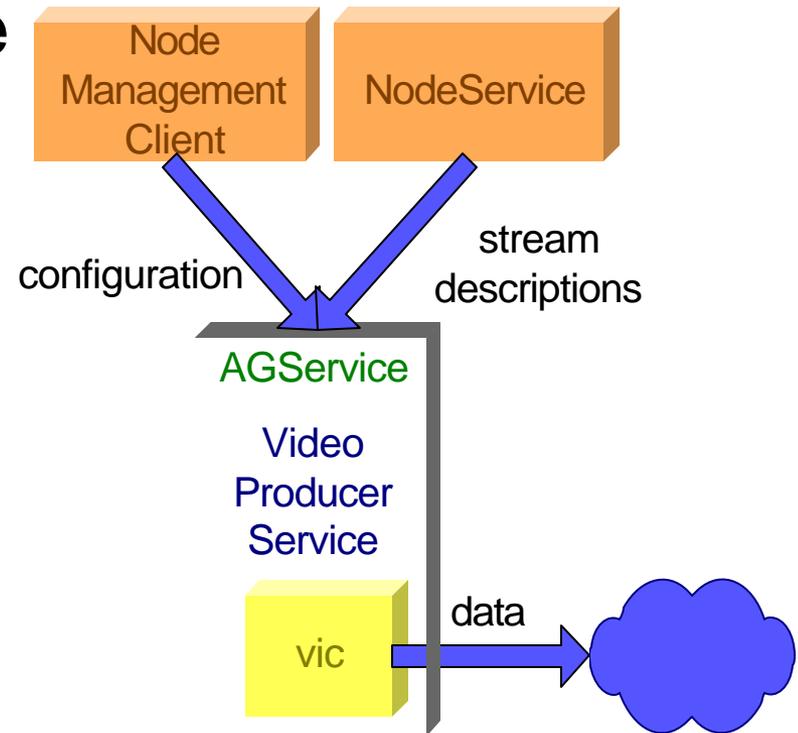
Specialization of AGService.py

- Using Python
 - AGService base class implements most of the AGService interface
 - Minimum specialization includes the following methods:
 - `__init__` – to set configuration parameters for the Node Service
 - `Start` – to start underlying software



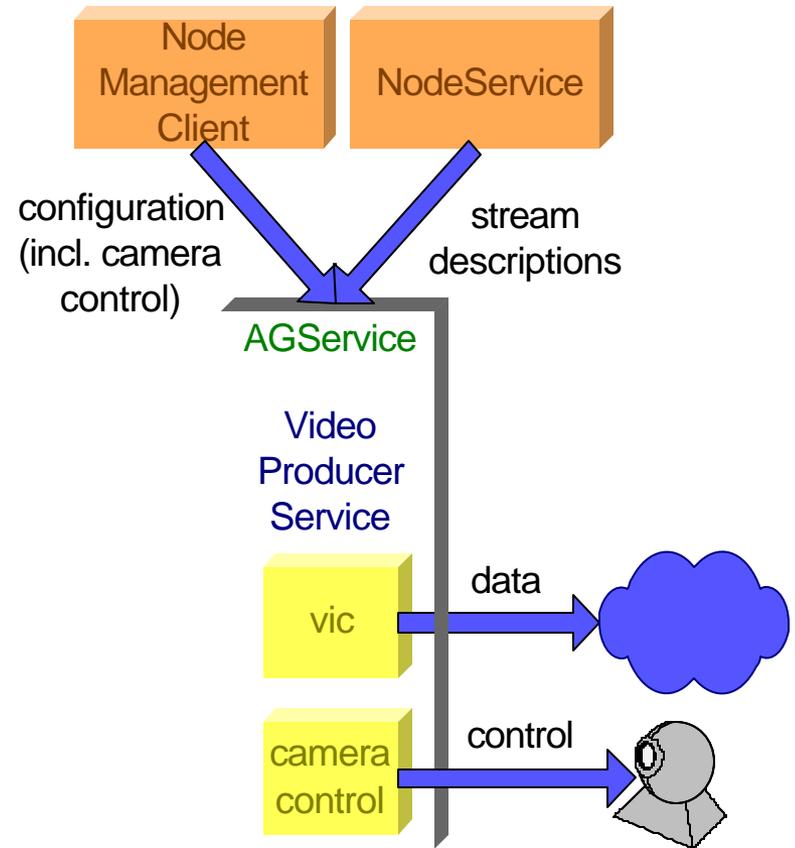
Review of VideoProducerService

- VideoProducerService implements the AGService interface
- Executes vic to transmit video stream
- Configurable through Node Management Client



Addition of camera control to VideoProducerService

- Add camera control parameters to configuration
- Integrate software to communicate commands to camera



Add Camera Control

Specialization of `__init__` to add camera control parameters

```
def __init__(self):
    # (snip)
    # serial port to control camera
    self.configuration["serial port"] =
        TextParameter("serial port", "")

    # camera id
    self.configuration["camera id"] = TextParameter("camera id", "")

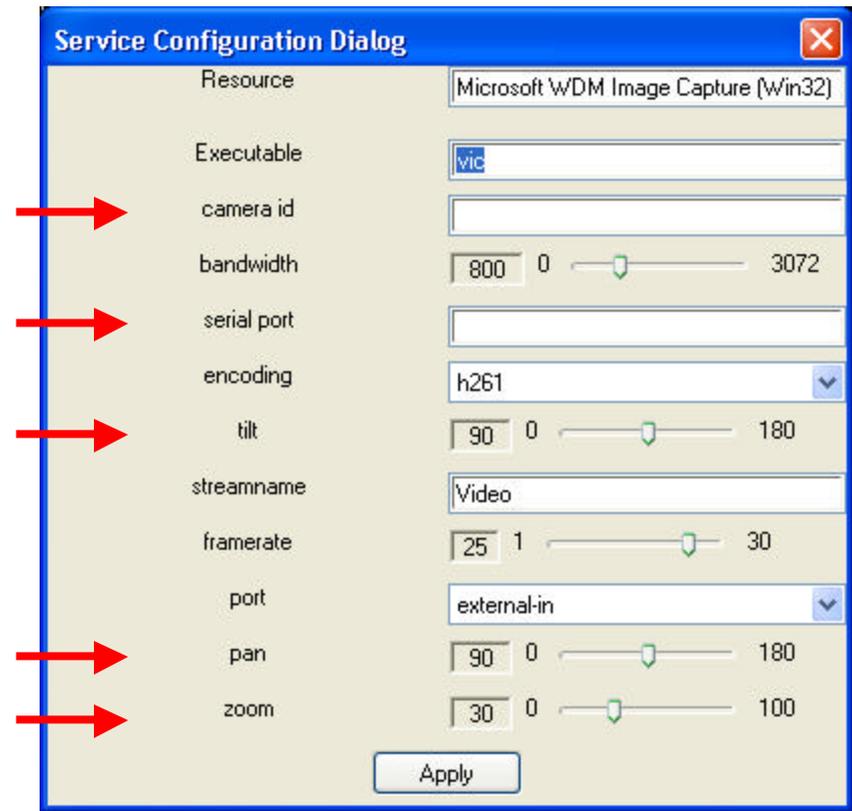
    # range parameters for zoom, pan, and tilt
    # e.g. RangeParameter(name,value,min,max)
    self.configuration["zoom"] = RangeParameter("zoom", 30, 0, 100)
    self.configuration["pan"] = RangeParameter("pan", 90, 0, 180)
    self.configuration["tilt"] = RangeParameter("tilt", 90, 0, 180)
```



Retreat 2003

Add Camera Control

- Camera control parameters appear on Service Configuration dialog
- Camera settings are stored with node configuration



Add Camera Control

Specialization of SetConfiguration

```
def SetConfiguration(self, configuration):  
    #(snip)  
    # update the camera appropriately  
    serialPort = self.configuration["serial port"]  
    cameraId = self.configuration["camera id"]  
    self.updateCamera(serialPort, cameraId, "zoom",  
        self.configuration["zoom"])  
    self.updateCamera(serialPort, cameraId, "pan",  
        self.configuration["pan"])  
    self.updateCamera(serialPort, cameraId, "tilt",  
        self.configuration["tilt"])
```



Summary

- (Node) services add to the capabilities of a node
- Services are simple to develop
- In Python, services are *simplistic* to develop

