What is GStreamer?

A library for building multimedia pipelines that is flexible and extensible while being cross platform. Comprehensive multimedia framework that handles audio and video processing tasks. Users can create different pipelines by combining elements together in chains, each element responsible for a different function. This creates a customizable amount of functionality from simple media playback to live feed streaming.

It is primarily written in C, and is known for its efficiency. The framework offers various bindings for languages such as **Python, C++,** and **Go.** GStreamer depends on plugin components for different media processing.

Some ways to use GStreamer include:

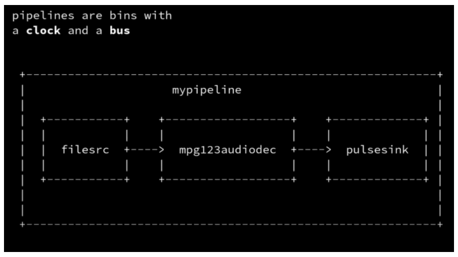
- HLS (HTTP Live Streaming) and call recording in video egress
- RTMP (Real Time Messaging Protocol) output implementations
- Thumbnail generation in SFU (Selective Format Unit)

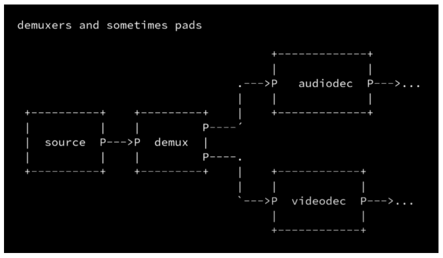
Example way of Implementation:

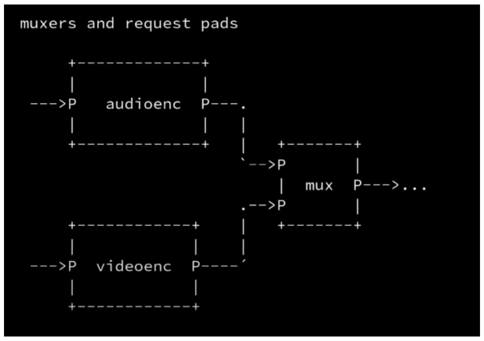
- gst-launch-1.0
  filesrc location=audio.mp3 !
  mpg123audiodec !
  audioconvert !
  osxaudiosink
- filesrc → mpg123audiodec → audioconvert → osxaudiosink

Elements:

basic units of functionality such as sources, filters, and sinks; basic building blocks

- Pads: Connection points; Sinkpad; Sourcepad
- Bins: an element that can contain other elements; are xzibit elements with ghostpads
- Pipelines: just different pathways of elements; top level bin that contains the entire processing pipeline.
- Example:
- 



- 

```
muxers and request pads

     +-------------+
     |             |
 --->P   audioenc  P---.
     |             |   |
     +-------------+   |   +-------+
                       `-->P       |
                           |  mux  P--->...
                       .-->P       |
     +------------+    |   +-------+
     |            |    |
 --->P  videoenc  P----´
     |            |
     +------------+
```

- Clocks are used to synchronize different elements together such as audio and video together in a time frame.

Installation:

Linux:

1. `sudo apt-get update`
2. `sudo apt-get install gstreamer1.0-tools gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly gstreamer1.0-plugins-bad`

On macOS:

1. `brew install gst-plugins-base gst-plugins-good gst-plugins-ugly gst-plugins-bad`

Digital Video Concepts:

- The procedure is called pipeline
- Steps are called elements
-

Python

Python Steps:

1. Give Gstreamer a string with the names of each element you want in your pipeline separated by exclamation marks
   a. Gstreamer takes care of creating these elements and attaching them together; making a pipeline
2. import gi
3. gi.require_version(“Gst”,”1.0”)
4. from gi.repository import Gst

Criteria met:

- Python has official bindings with GStreamer
- Has excellent popular GUI libraries such as PyQt or GTK
- Can combine with PyQT or GTK libraries to view Gstreamer stream

Python Gstreamer Binding Options:

1. **gst-python**
- Requires good knowledge of Gstreamer concepts.
- Is the official bindings with GStreamer

2. **PyGObject**
- Python binding for **GObject-based** libraries.
- Both GTK and Gstreamer are GObject based.

PyQt vs GTK (GUI libraries):

- GTK offers seamless embedding of Gstreamer video stream through Gtk.DrawingArea as compared to PyQT which requires some manual setup.
- PyQt is more complex, useful if you have previous experience. GTK is easier to work with gst-python, and has natural integration due to both being part of the GObject ecosystem.
- Lower learning curve with GTK due to native support of multimedia integration.
- GTK is the better choice for a quicker and easier integration of GUI with GStreamer.

What is Golang?

GoLang or Go developed by Google is a static compiled language that is simple, efficient, and has a strong support. It is well suited for developing scalable network applications such as multimedia processing.

Me when I find out Go is not one of the languages officially supported by GStreamer:

GoLang Gstreamer Binding Options:

- Gstreamer Go Bindings
- Go-GObject Bindings
- go-gst
- Qt (Additional Option)

Criteria met:
- Has bindings with GStreamer
- Compatible with GUI libraries such as GoQt and GTK
- Can view Gstreamer stream through QtGStreamer bindings

1. go-gst (Built upon Gstreamer Go Bindings)

- **About:**

  Using cgo, these are direct bindings to the GStreamer C library, however they expose much of the GStreamer API to Go. Built upon GStreamer Go Bindings, and is one of the most popular and well maintained GStreamer bindings for Go.
- **Pros:**
  - Works directly with GStreamer 1.x.
  - Good for building custom pipelines and handling multimedia data in Go.
  - High level, Go-friendly API
  - Actively maintained
  - General Purpose GStreamer in Go
- **Cons:**
  - Low-level access to GStreamer functionality
  - Higher level of debugging difficulty
  - Limited Documentation
  - Higher learning curve and understanding of GStreamer's architecture
  - Newer and has lesser features with fewer libraries and smaller ecosystem.
- **Implementation:**
  - Requirements:
    - `cgo`: Must set `CGO_ENABLED=1` in your environment when binding
    - `gcc` and `pkg-config`
  - Installation:
    - `go get github.com/tinyzimmer/go-gst`

2. Qt

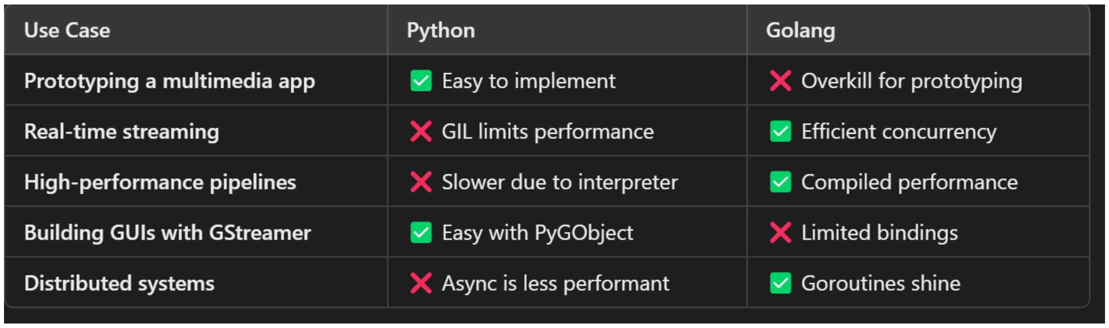
- **About:**

  C++ framework - can be used with other languages such as Go, JavaScript and Python through Qt bindings (not as complex with Go alone). Qt does not natively support GO, therefore some libraries are required to implement it.
- **Usage:**
  - Install Qt:
    - `sudo apt-get install qtbase5-dev qt5-qmake`
  - Install qt-go library
    - `go get -u github.com/therecipe/qt`
  - Setup environment with `$(go env GOPATH)/src/github.com/therecipe/qt/cmd/qtsetup`
  - Run the Go application with `go run main.go`
  - Other ways to connect:
    - Using CGo with Native Qt C++ code
    - **Using QML with Go Backend**
- **Additional features:**
  - Runs on many platforms
  - Very flexible and customisable

Evaluation:

While Qt can be integrated with both Go and Python, it would require **Qt bindings**, which is not a straightforward approach, hence using **C++** for Qt can be considered an additional option. Qt is a full fledged application development framework. This may mean that it is a little harder to get the hang of, but it has a much larger ecosystem. At a later point, go-gst becomes challenging to build and expand off from with its narrow limitations. Overall, Qt is easier for general purpose applications and GUIs, while go-gst is better for specialized multimedia backends. When it comes to community and documentation, Qt has a larger support and documentation, making it easier to troubleshoot. In addition, Qt also allows for faster development, especially with UI heavy applications.

Python vs Golang

- Python has official bindings with GStreamer while Golang does not
- Golang is better for performance-critical applications when real-time processing is required with its speed capabilities.
    - Python is an implemented language, making it slower than GoLang
    - GStreamer pipelines implemented in python would further slow it down, especially for heavier tasks.
    - Golang is compiled, giving it better performance for CPU related tasks.
- Python is more beginner friendly.
- Python has better community support.
- Python is easier for debugging.
- While both of them share libraries such as Qt and GTK, most of Golang's Gstreamer bindings are underdeveloped and not known as compared to Python's Gstreamer bindings.
- 

Use Case	Python	Golang
Prototyping a multimedia app	✓ Easy to implement	✗ Overkill for prototyping
Real-time streaming	✗ GIL limits performance	✓ Efficient concurrency
High-performance pipelines	✗ Slower due to interpreter	✓ Compiled performance
Building GUIs with GStreamer	✓ Easy with PyGObject	✗ Limited bindings
Distributed systems	✗ Async is less performant	✓ Goroutines shine

Additional Note:

Rida and Syed started the report hyping their own assigned language up and ended preferring the opposite language by the end of this report.