# Google Summer of Code 2016

https://g.co/gsoc

Scratch2Catrobat Converter – Final Presentation

*by Ralph Samer*

# What is Scratch2Catrobat Converter?

- A compiler tool for converting Scratch programs into Catrobat programs:
  - converts Scratch code files (JSON) into Catrobat code files (XML)
  - checks for audio & image files that are not playable/displayable on Android and converts them into other compatible media formats (e.g. SVG files are converted to PNG files)

# Major goals


- **Fill the gap** between Scratch system and Catrobat project
  - it empowers kids to run their self-made Scratch programs on their own phone
  - That means, users can create programs on their PC/notebook
    - ... and then they can open and run converted programs on their smartphones/tablets

- **Fast conversions**

  users don't have much time to wait…

- **Produce accurate results**
  - … but Scratch programs are designed to run on larger displays
  - ... but some Scratch blocks do not behave the same way as their Catrobat equivalents do:
    - all this forces us to use tricky workarounds

# Overview of Scratch2Catrobat project


Consists of the following 3 different applications:

- **Scratch2Catrobat Converter** (java & python code mixed → Jython interpreter)
- **Source Code Filter** (pure Java implementation)
- **Web Application** (pure Python implementation)

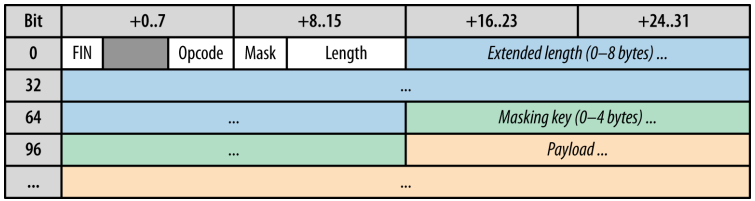
# Pocket Code - Integrate S2C


- **Client side:**
  - Android WebSocket client integrated into Catroid
    - talks to WebSocket server by using its own protocol

  - **In contrast to HTTP, communication is asynchronous here!**
    (i.e. server may reply shortly after or some time later
    e.g. when a task is complete)

# Pocket Code - Integrate S2C

- **Protocol:**
  - Consists of:
    - Commands
    - Messages (base/info & job messages)
  - JSON formatted and put into payload field of a WebSocket message/frame
  - WebSocket clients can send commands to server
  - Server replies with base/info/job messages

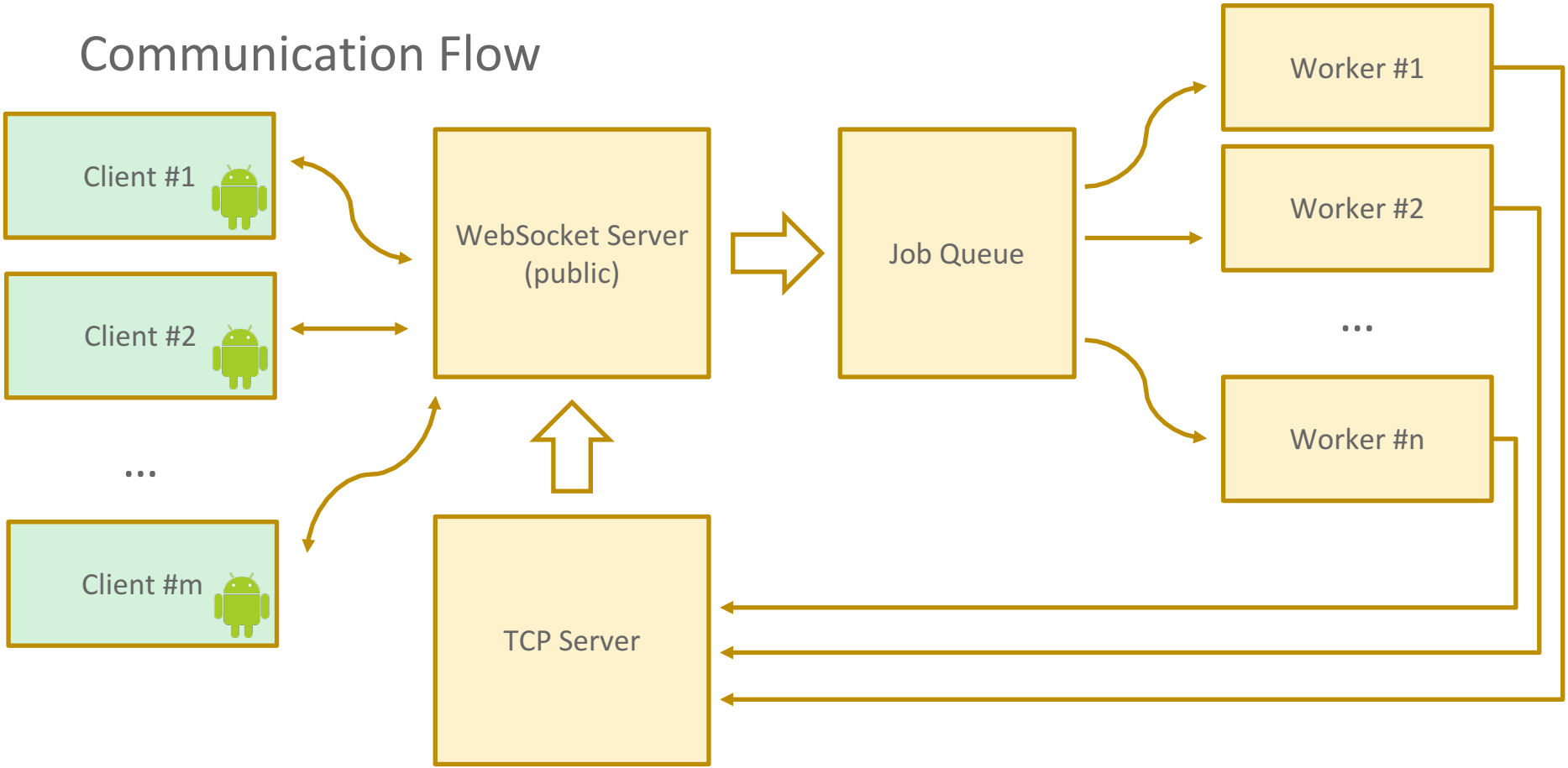



*Source:*
*http://chimera.labs.oreilly.com/books/1230000000545/ch17.html*

# Scratch2Catrobat Converter Server


- **Server side:**
  - WebSocket Application Server
    - publicly accessible by clients
    - puts new jobs into queue
  - Job Queue System
  - TCP Server
    - communicates with workers
  - Workers
    - are responsible for the conversion process on the server
    - fetch jobs from queue and process them
    - a TCP client is integrated in order to connect to the (central) TCP server

# Communication Flow



# Tasks & Mile stones (I)


| Issue-ID | Title | Hours spend working on task |
|---|---|---|
| CAT-1866 | Integrate Scratch search into Catroid, voice search, Async task, cache search results, image caching, tests, etc. | **~40h** |
| CAT-1923 | Activity for showing details of Scratch programs, custom designed layout, Async task, cache search results, image caching, tests, etc. | **~25h** |
| CAT-1932 | Websocket Client & communication protocol | **~105h** |
| CAT-1859 | Main user interface (list views for search, sliding up panel area, etc.) | **~30h** |

# Tasks & Mile stones (II)


| Issue-ID | Title | Hours spend working on task |
|---|---|---|
| STCC-28 | Implementation of WebSocket application & API for Websocket clients (Server) | ~120h |
| STCC-27 | Proxy for fetching Scratch project details (Server) | ~10h |
| STCC-3 | Fix image size problem of converted programs (Converter) | ~5h |
| STCC-51 | Update serializer and data model classes (Converter) | ~10h |
| STCC-29 | Workaround for distance-to block (Converter) | ~5h |
| STCC-47 | Further automate SourceCodeFilter (Source Code Filter) | ~10h |

# Challenges


## Performance

- fast and resource efficient
- asynchronous communication
- handle low-bandwidth & long latency connections
- reduce traffic (e.g. by using lightweight data formats like JSON)

## Scalability

- Web server optimized for handling thousands of WebSocket connections
  → tries to solve C10k problem

- Worker instances:
  - are supposed to run on **different servers**
  - can be **dynamically added/removed** during run time

# Difficulties/Lessons learned


- Modularize code
  - mock & test only interfaces instead of classes!
  - design patterns (e.g. observer pattern to notify activities when new messages arrive)
  - avoid instantiating classes within other classes (use dependency injection, factory classes/methods, …)
  - don’t reinvent the wheel! (use existing libraries for everyday tasks)

- Merging
  - Overlapping changes introduced by other GSoC projects (e.g. scenes, new bricks, …)
  - New folder structure

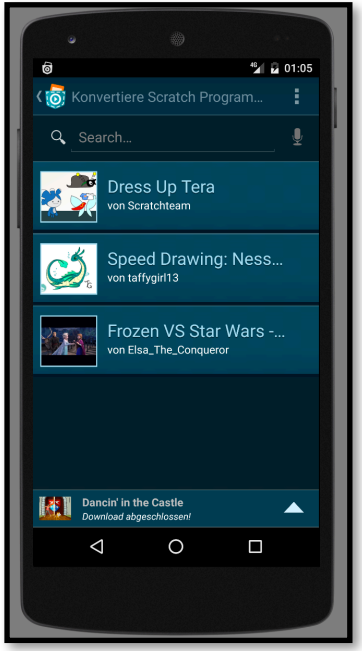
# Summary of open issues and remaining tasks


- On Catroid-side:
  - Pagination
  - Extend history view in Sliding Up Panel area
  - Verbose mode for debugging (optional)

- On Server-side:
  - Migrate to SSL/TLS
  - Authentication (Google+ Login, Facebook Login, LDAP, …)

- For Converter:
  - Support new Catrobat-Bricks → see other GSoC projects
    - User Bricks (Stefan Jaindl)
    - Go-to-, Distance-to-, Current time-, Stop-script brick, … (Robert Riedl)

Now it’s time for a live demo…



# Questions


Thank you for your attention!