Maximizing Throughput on Multicore Systems with Erlang


UBIQUITI
NETWORKS


RESTful http
server
handling audio
configurations

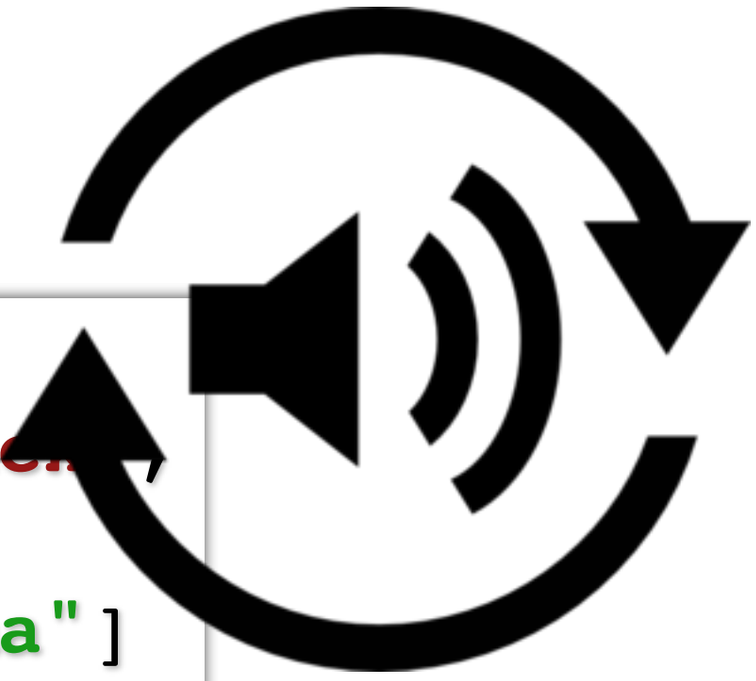

UBIQUITI


```
{
 “text": “Hi, it’s Jim. Call me back”,
 “voice_actor": "en_US_male",
 “output_files": [“wav","mp3","pcma"]
}
```

HTTP Server

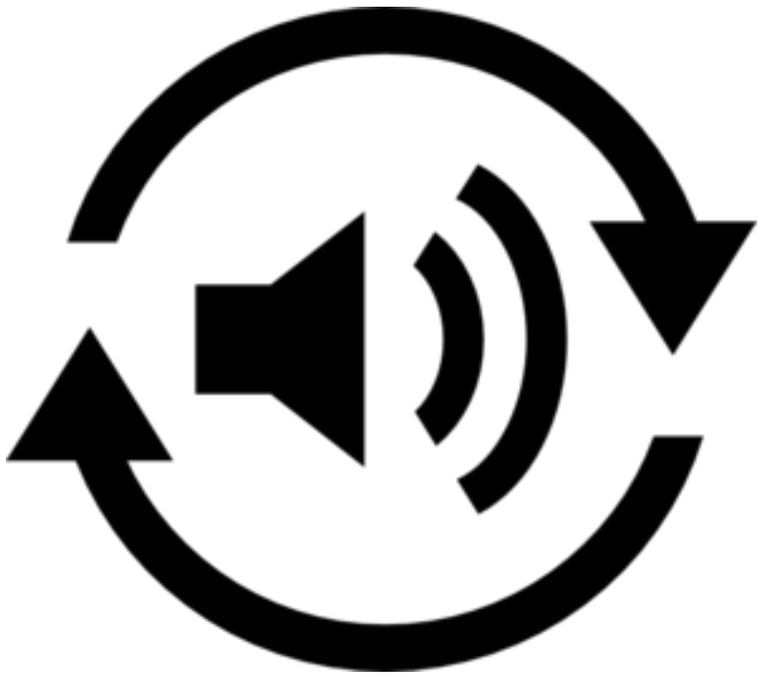
```
{
 “text": “Hi, it’s Jim. Call me back”,
 “voice_actor": "en_US_male",
 “output_files": [“wav","mp3","pcma"]
}
```

HTTP Server

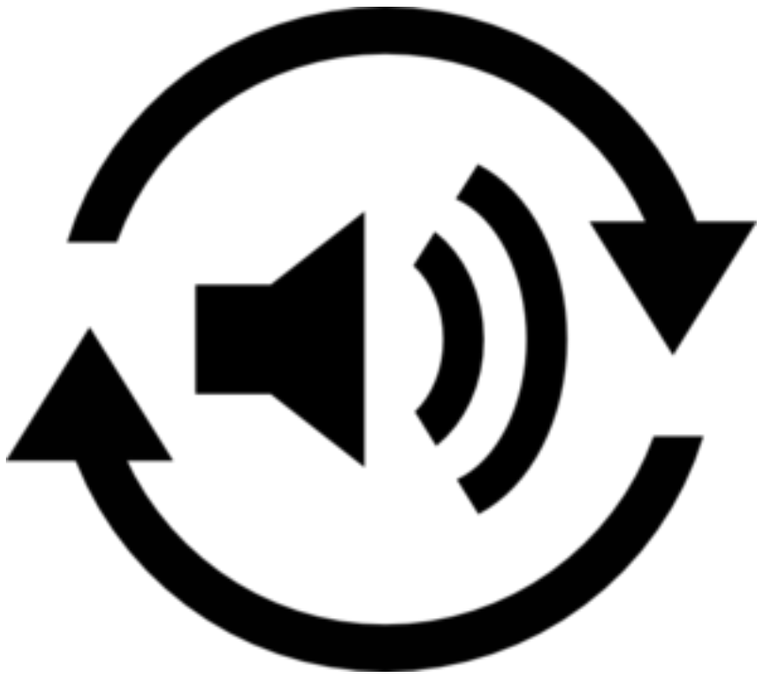


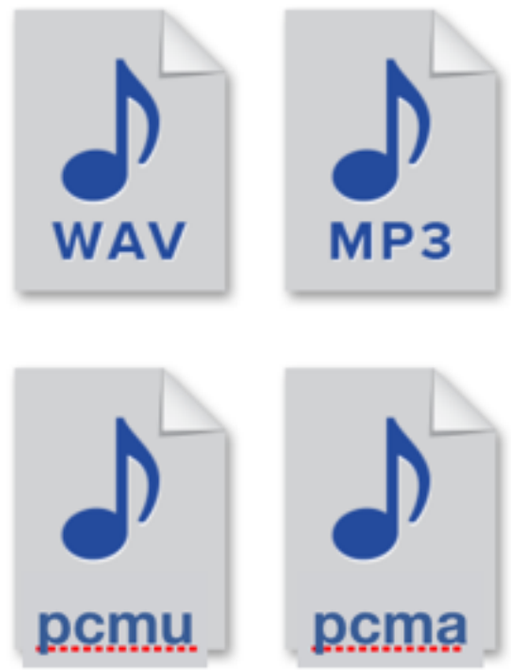
```
{
 “text": “Hi, it’s Jim. Call me back.”,
 “voice_actor": "en_US_male",
 “output_files": [“wav","mp3","pcma"]
}
```

HTTP Server

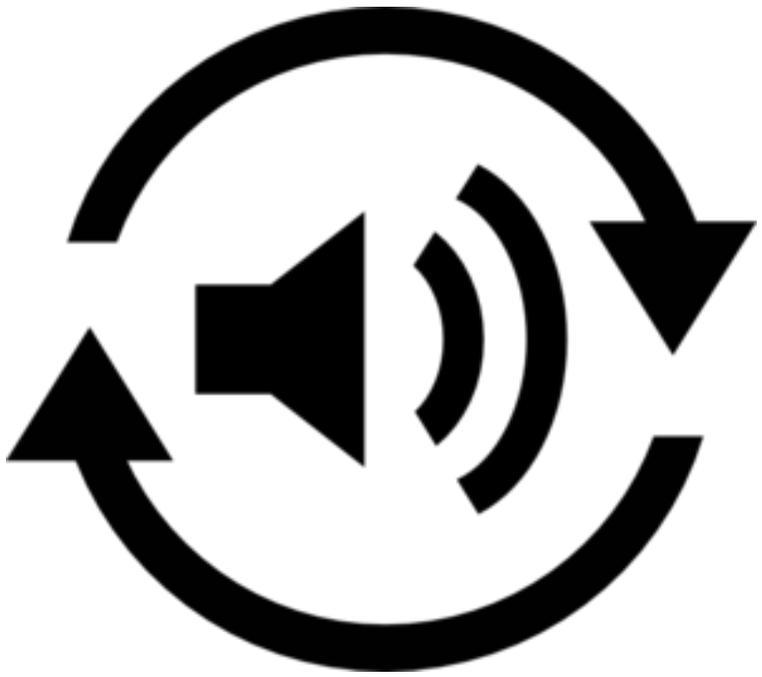


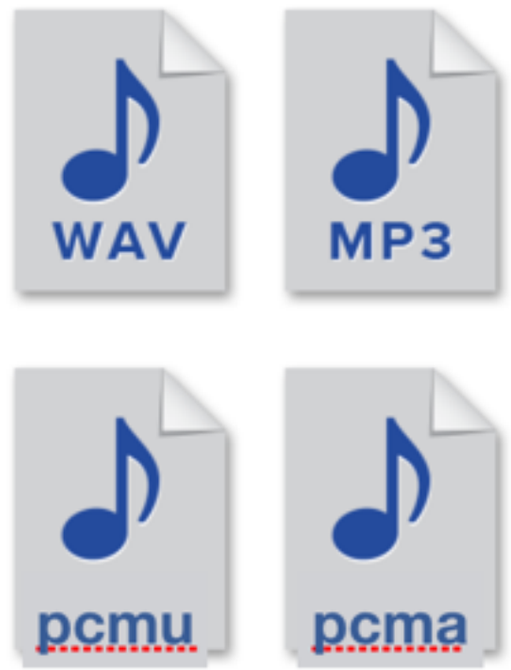
HTTP Server



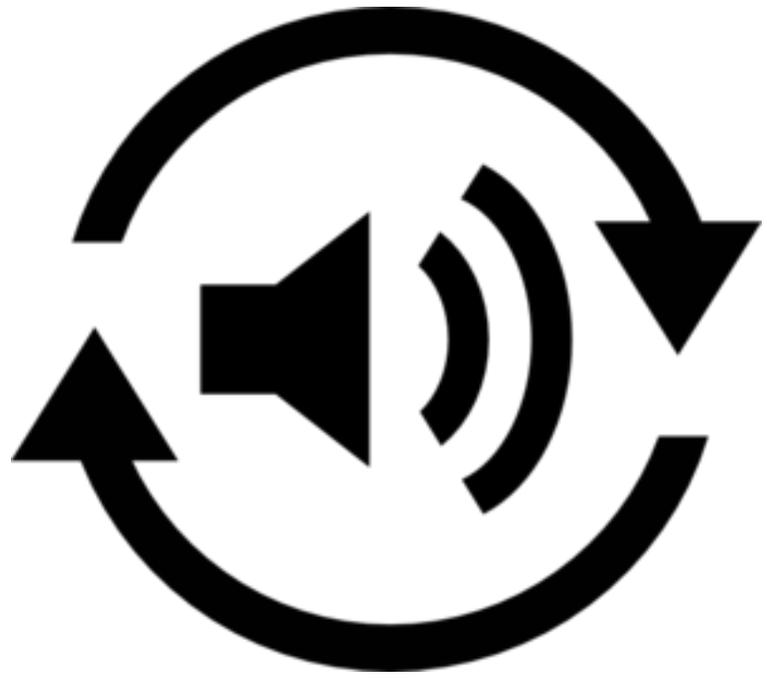


HTTP Server

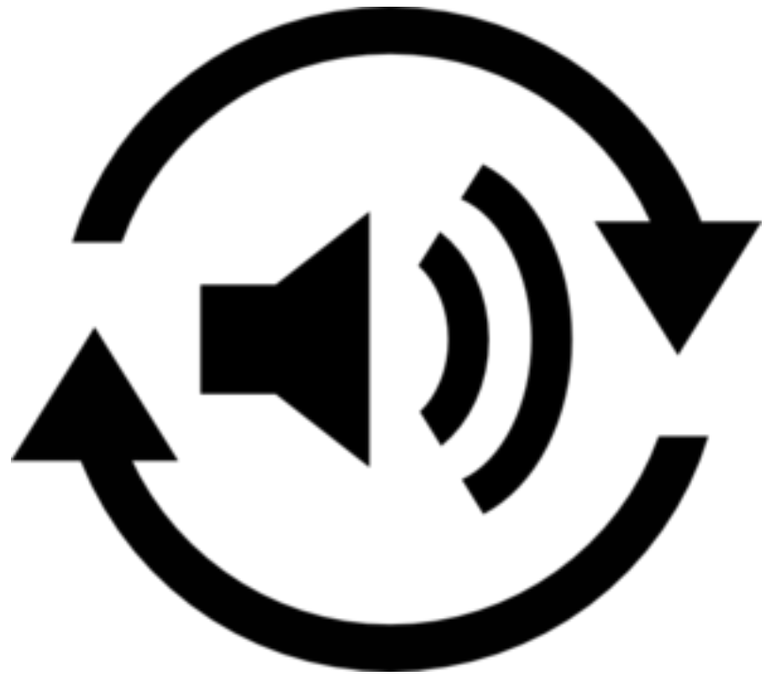




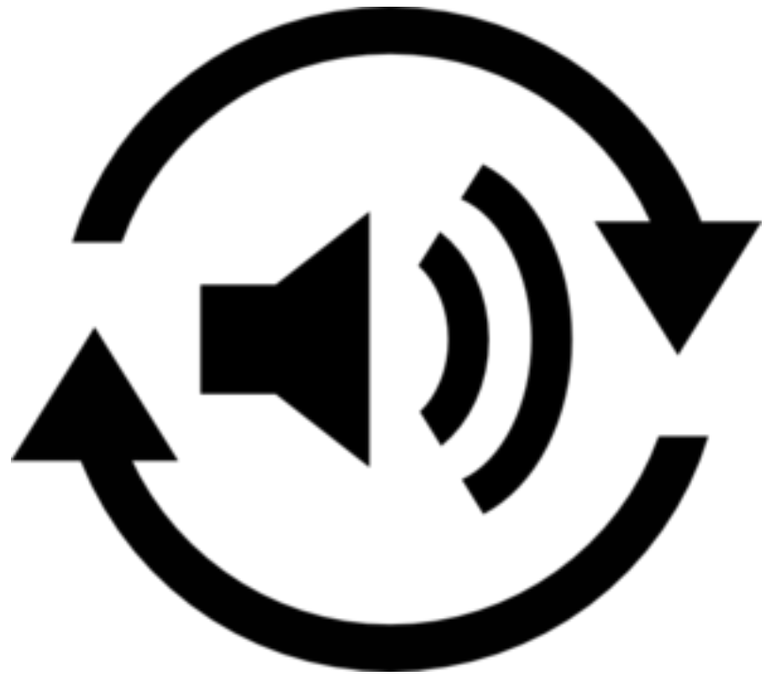
HTTP Server



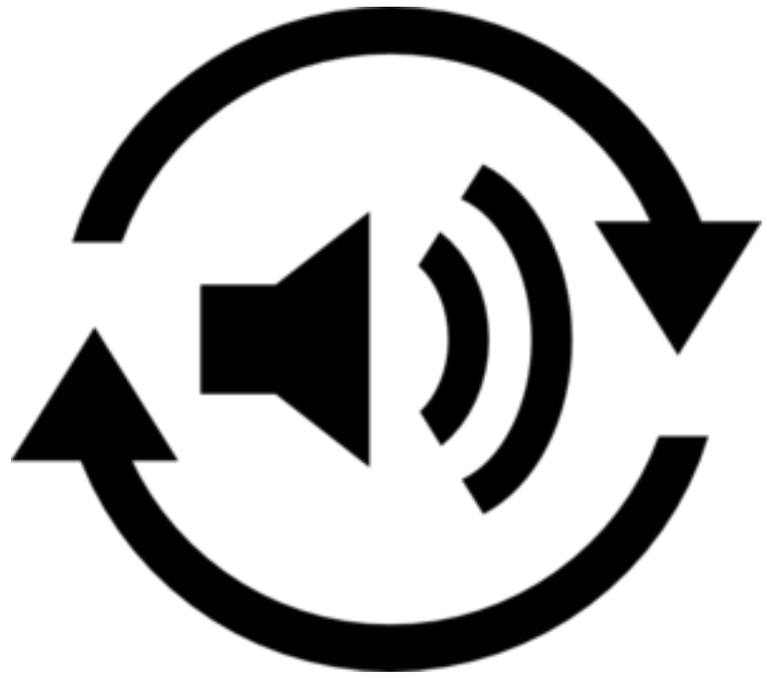
HTTP Server



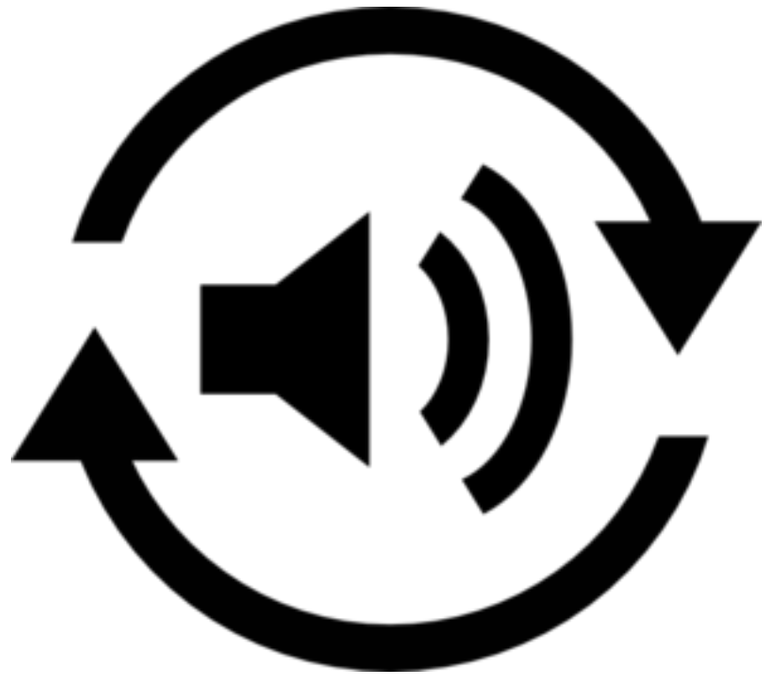
HTTP Server

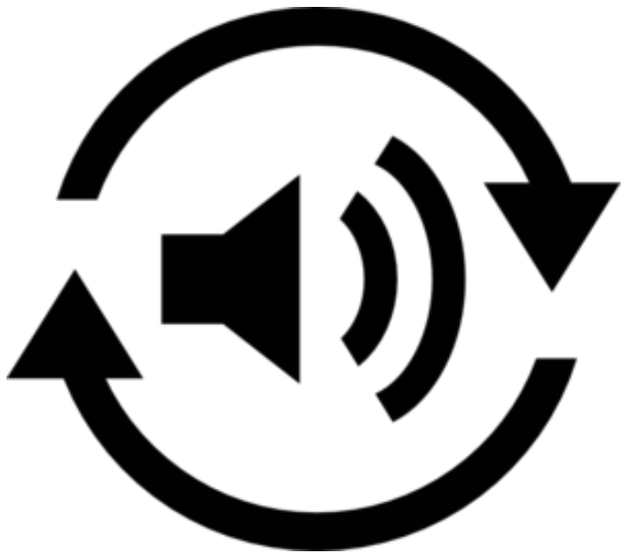


HTTP Server

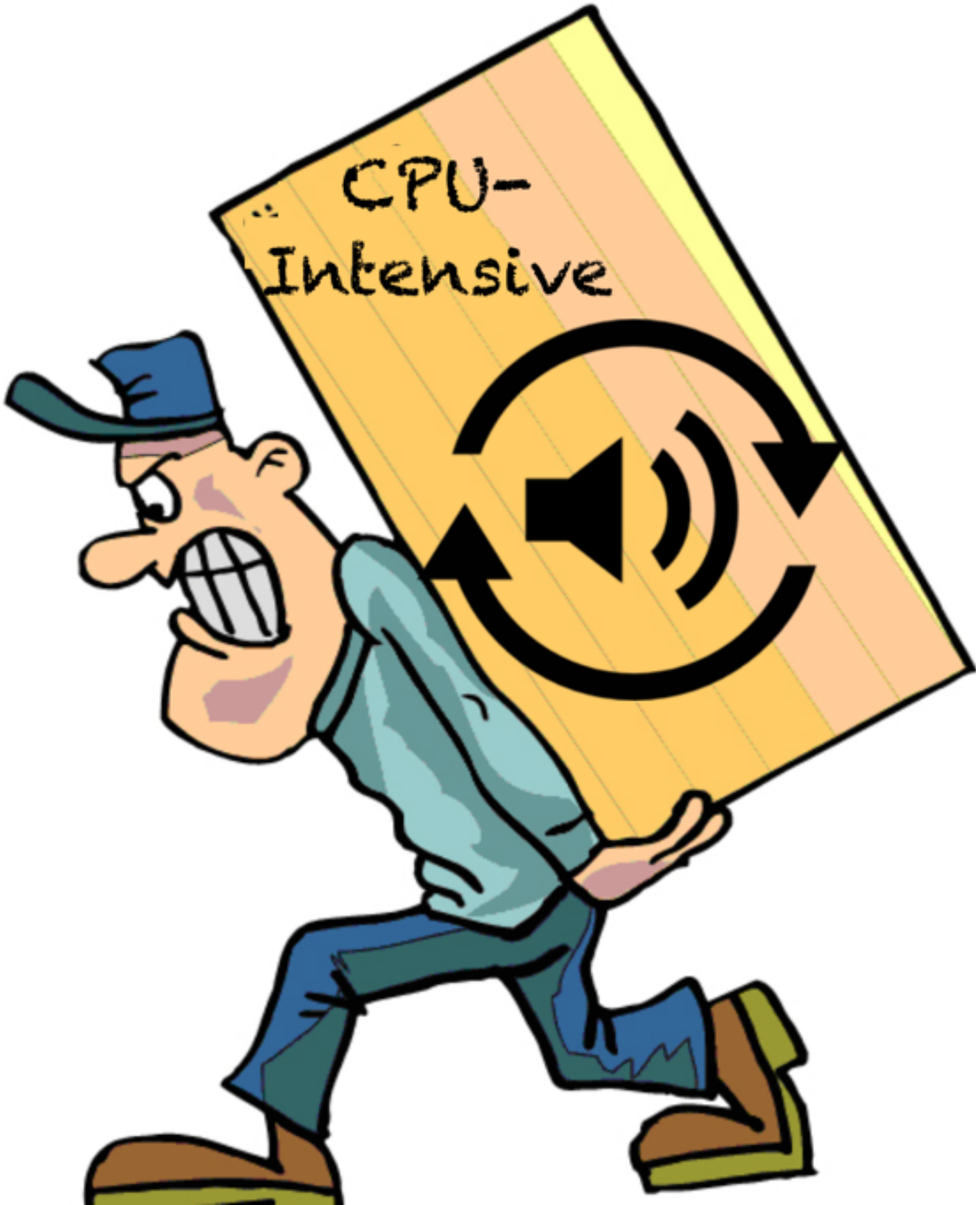


HTTP Server

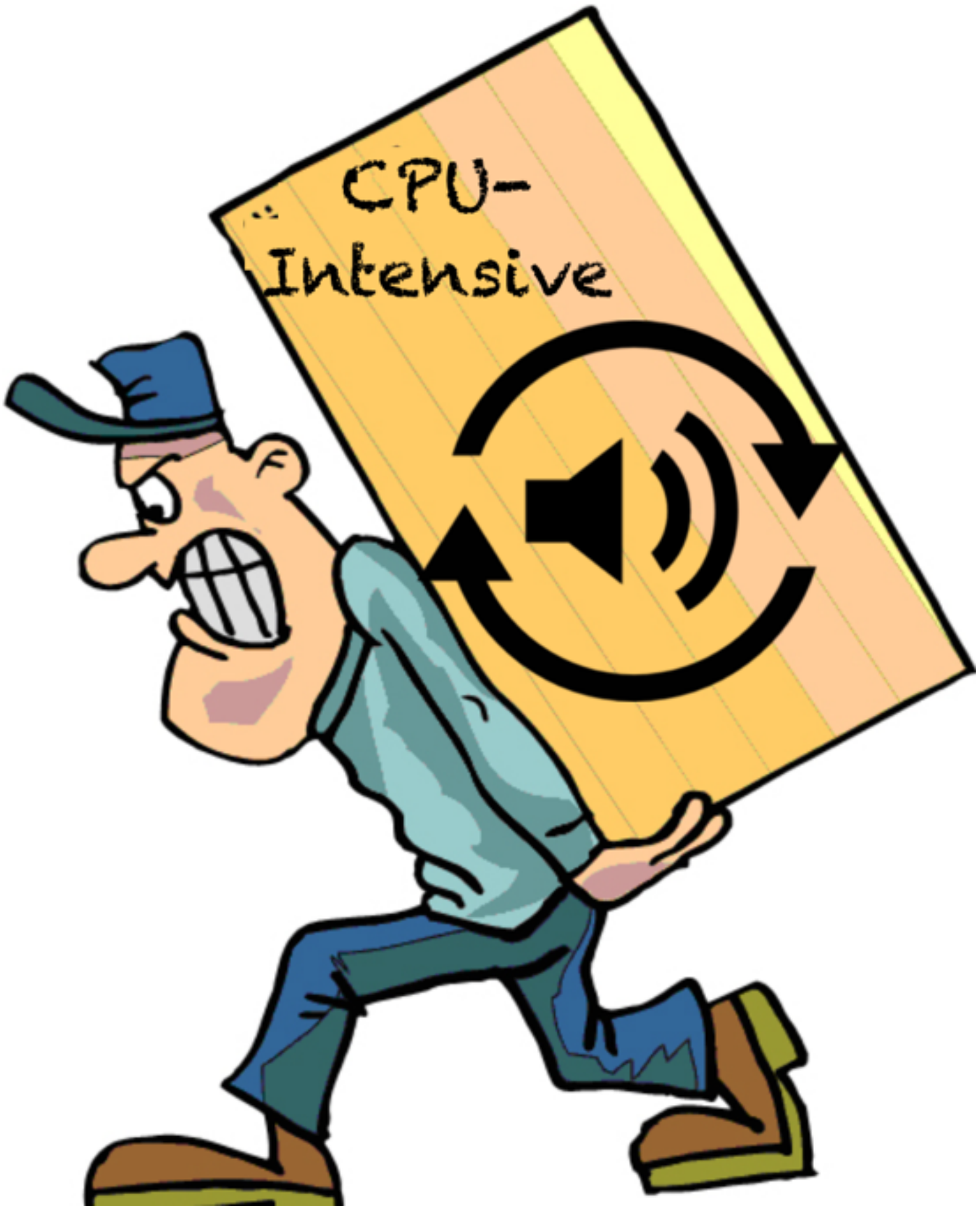





S3


CPU-
Intensive


S3


CPU-
Intensive


S3

Our Approach in Erlang

Finite State Machines

Finite State Machines

```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

Finite State Machines

```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

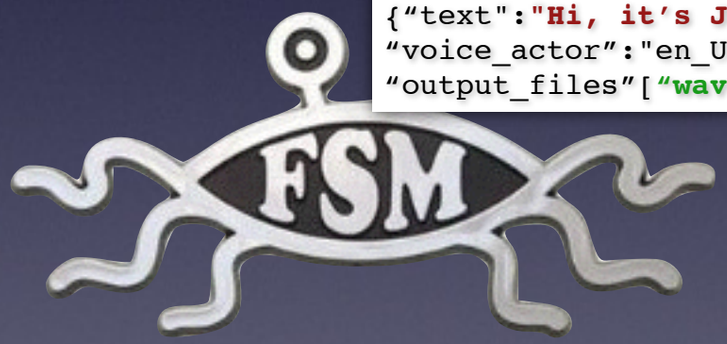

```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

Finite State Machines


```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines




```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

Finite State Machines


```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines


```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

Finite State Machines

{“text":"Hi, it’s
“voice_actor”:"en
“output_files”[“

Finite State Machines

{“text":"Hi, it’s
“voice_actor”:"en
“output_files”[“

Finite State Machines

Finite State Machines

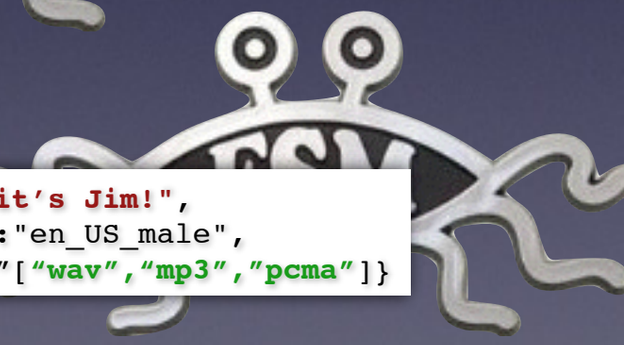
{“text":"Hi, it’s
“voice_actor”:"en
“output_files”[“v

Finite State Machines

Finite State Machines

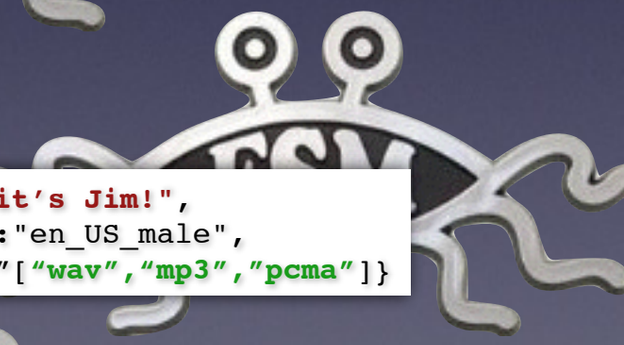
Finite State Machines

Finite State Machines




```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines




```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

Finite State Machines


```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

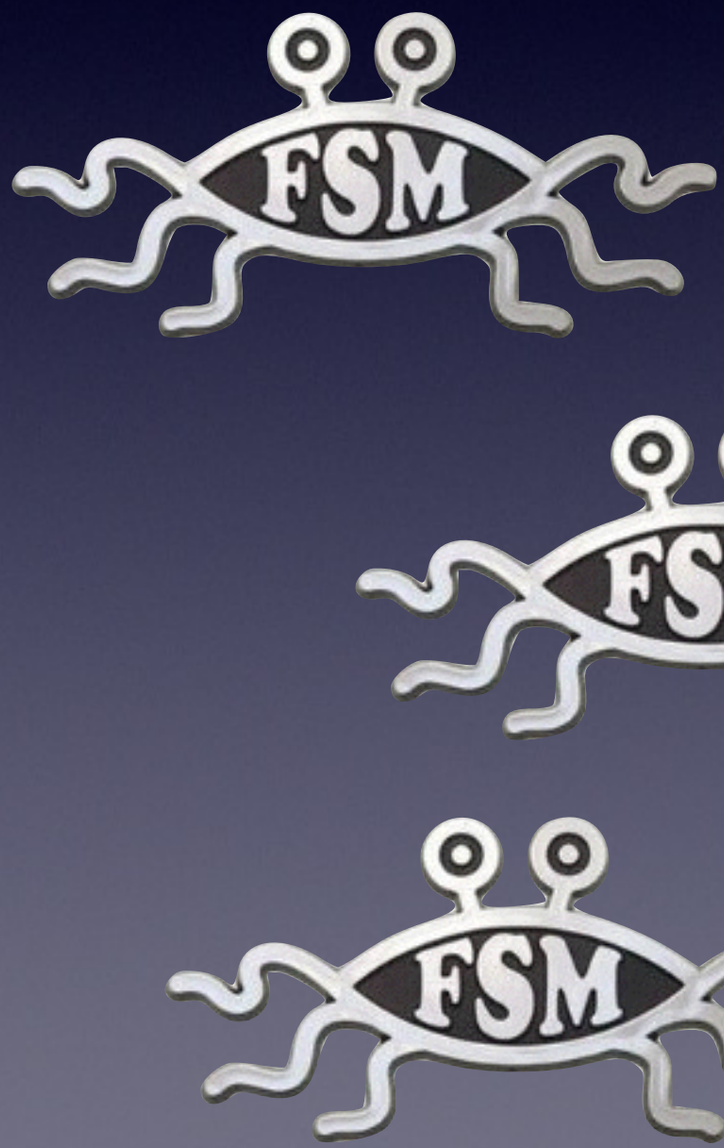

```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

Finite State Machines

Finite State Machines

Finite State Machines

```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

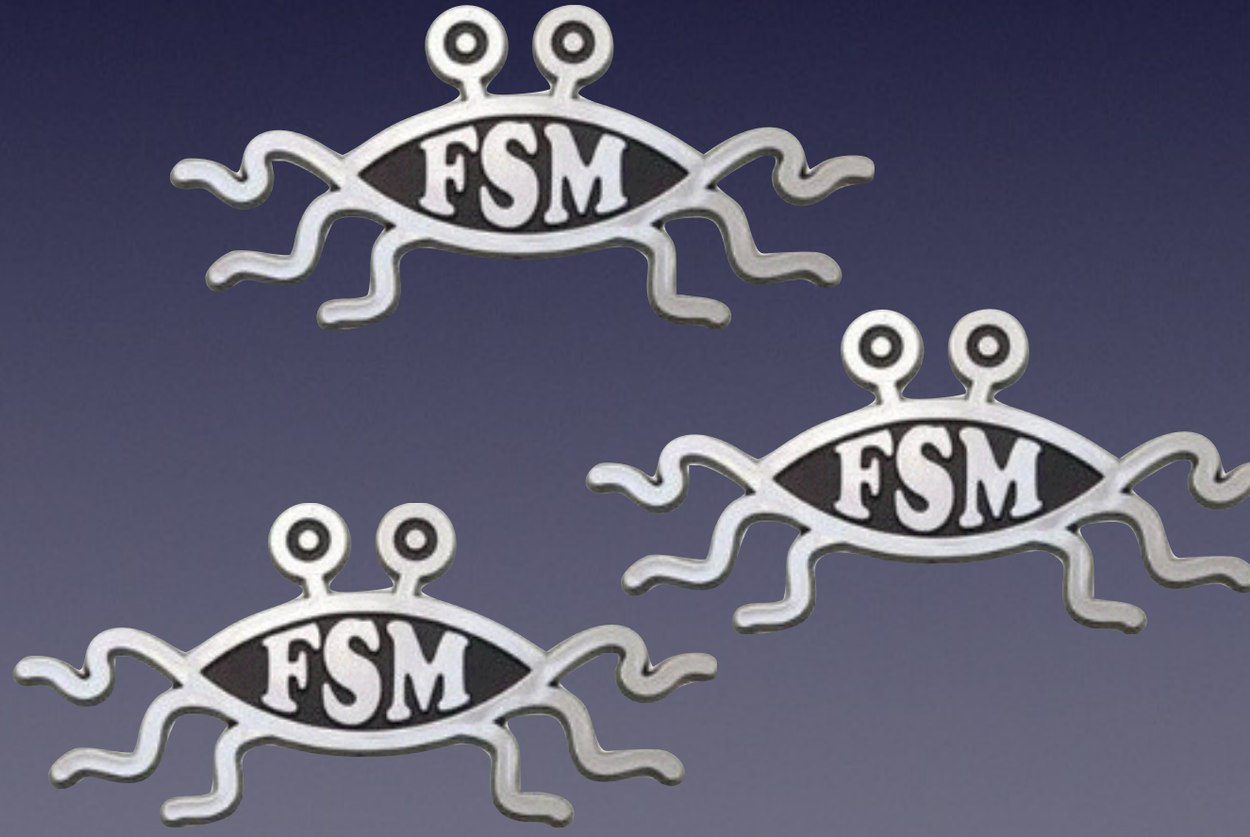


Finite State Machines

```
{“text":"Hi, it’s Jim!",
“voice_actor”:"en_US_male",
“output_files”[“wav”,“mp3”,”pcma”]}
```

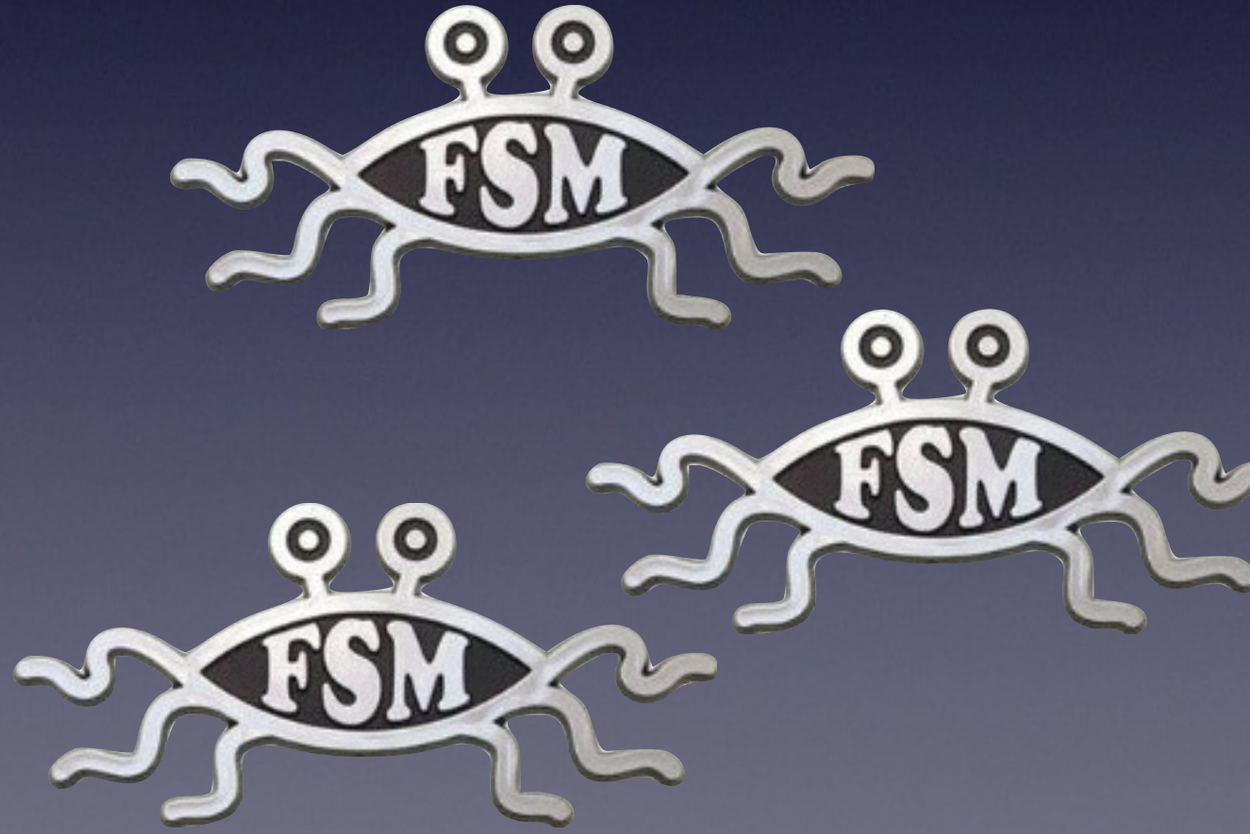
Finite State Machines

Finite State Machines

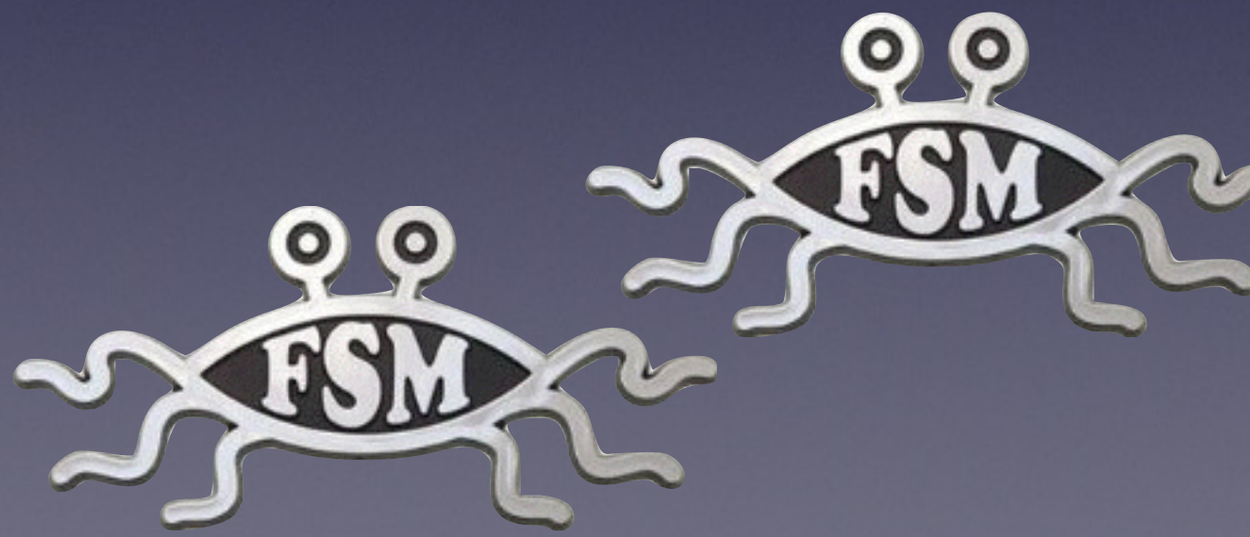


Finite State Machines

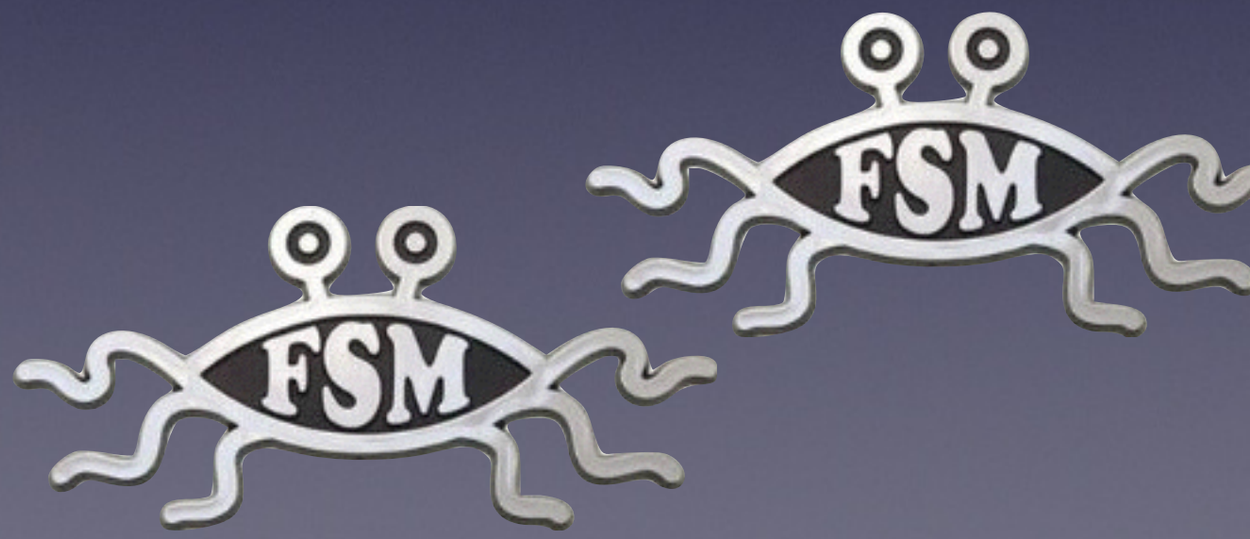
Finite State Machines



Finite State Machines



Finite State Machines



Finite State Machines

Finite State Machines

Erlang Processes are CHEAP

327 words when spawned including a heap of 233 words

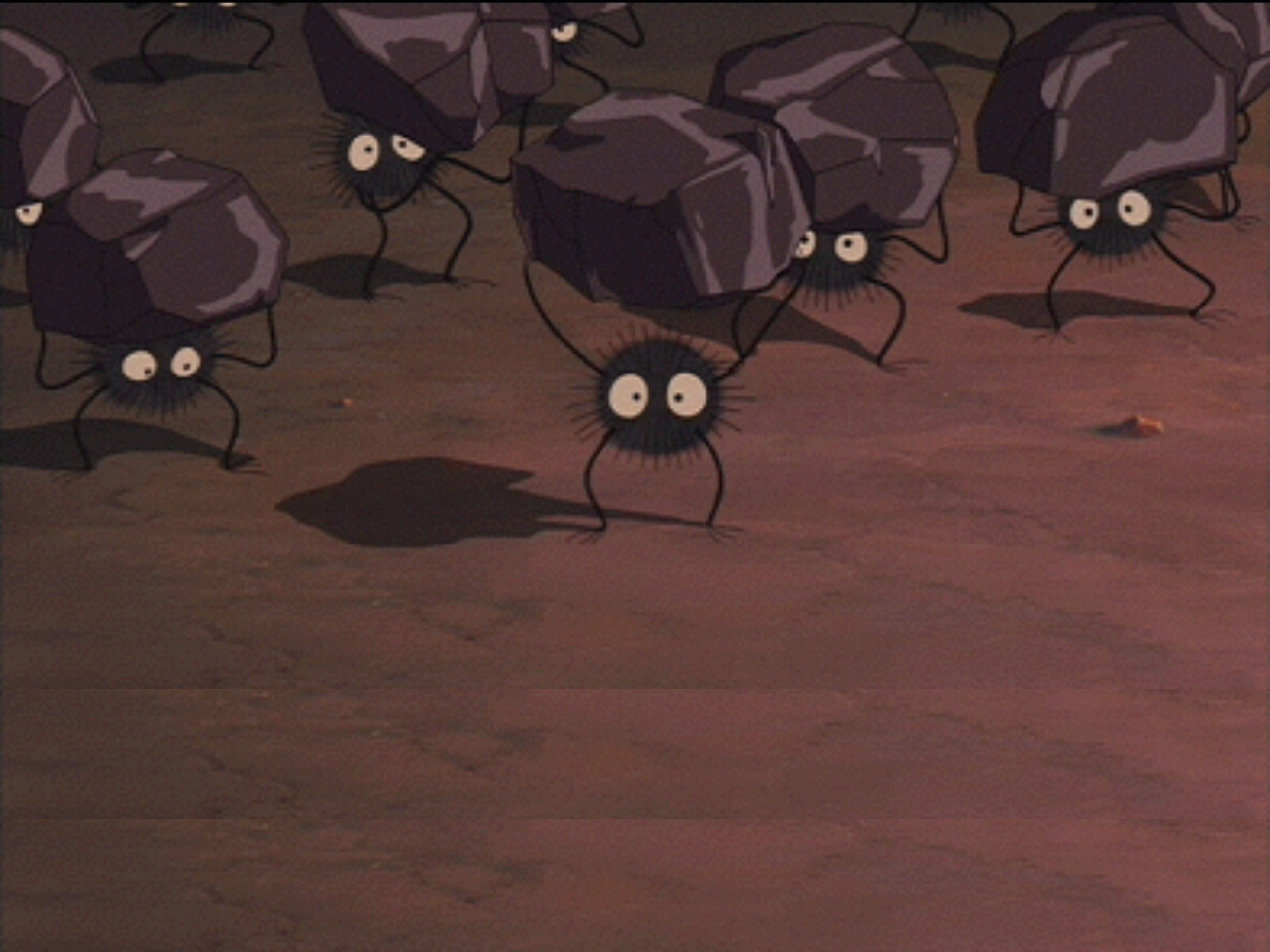
use existing C executables

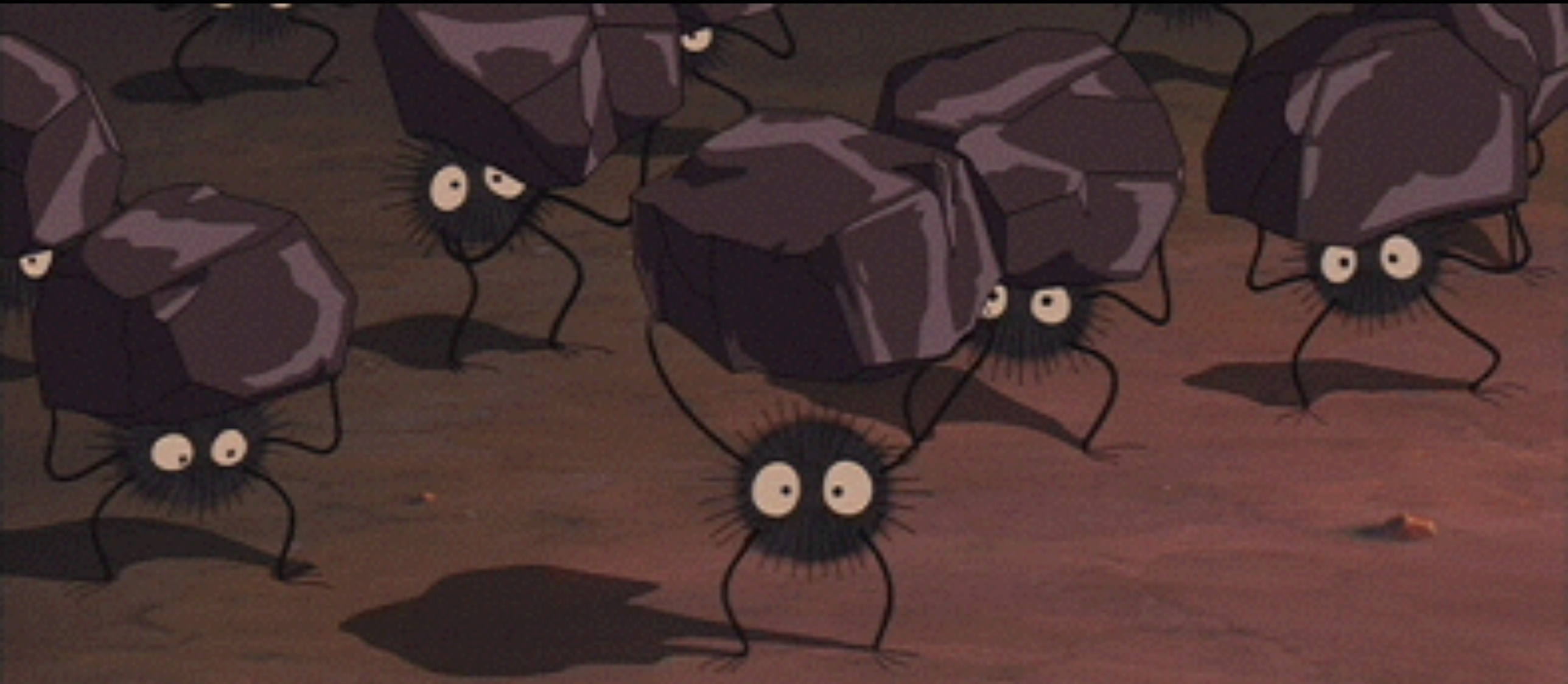
use existing C executables

- CEPSTRAL

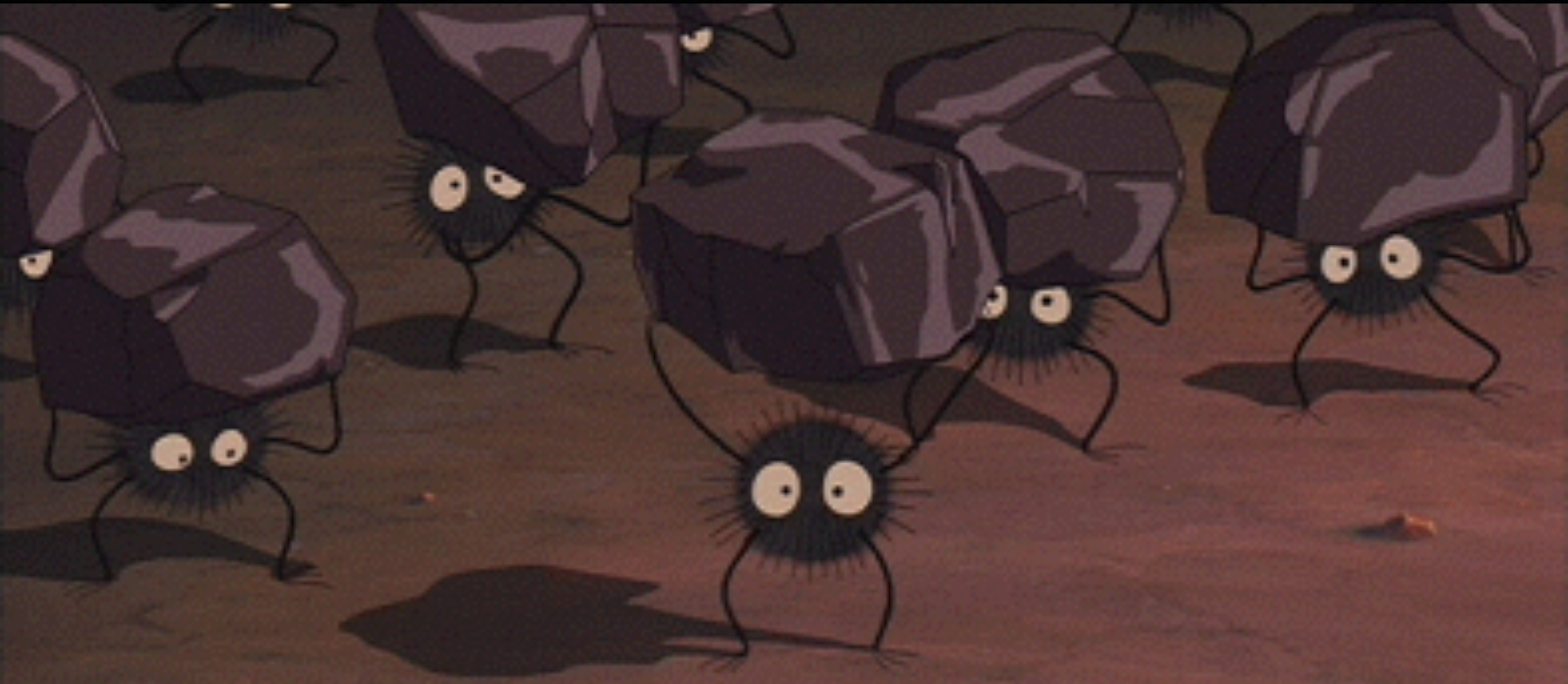
use existing C executables

- CEPSTRAL
- SOX





- Need to implement your own CPU-intensive task?



- Need to implement your own CPU-intensive task?

- Write a single unit of work in C

“Think of Erlang as a manager, not a worker”

–Garrett Smith


S3

S3url =

S3url =

https://s3.amazonaws.com:443/mybucket/my/audio/101/file1.wav?AWSAccessKeyId=KEY12345&Expires=5432138&Signature=MySignature

S3url =

https://s3.amazonaws.com:443/mybucket/my/audio/101/file1.wav?AWSAccessKeyId=KEY12345&Expires=5432138&Signature=MySignature

AudioBin=

S3url =

https://s3.amazonaws.com:443/mybucket/my/audio/101/file1.wav?AWSAccessKeyId=KEY12345&Expires=5432138&Signature=MySignature

AudioBin=


S3


```
spawn_link(
 ?MODULE,
 send_s3_request,
 [S3url,AudioBin,self()]).
```


```
spawn_link(
 ?MODULE,
 send_s3_request,
 [S3url,AudioBin,self()]).
```


```
spawn_link(
 ?MODULE,
 send_s3_request,
 [S3url,AudioBin,self()]).
```

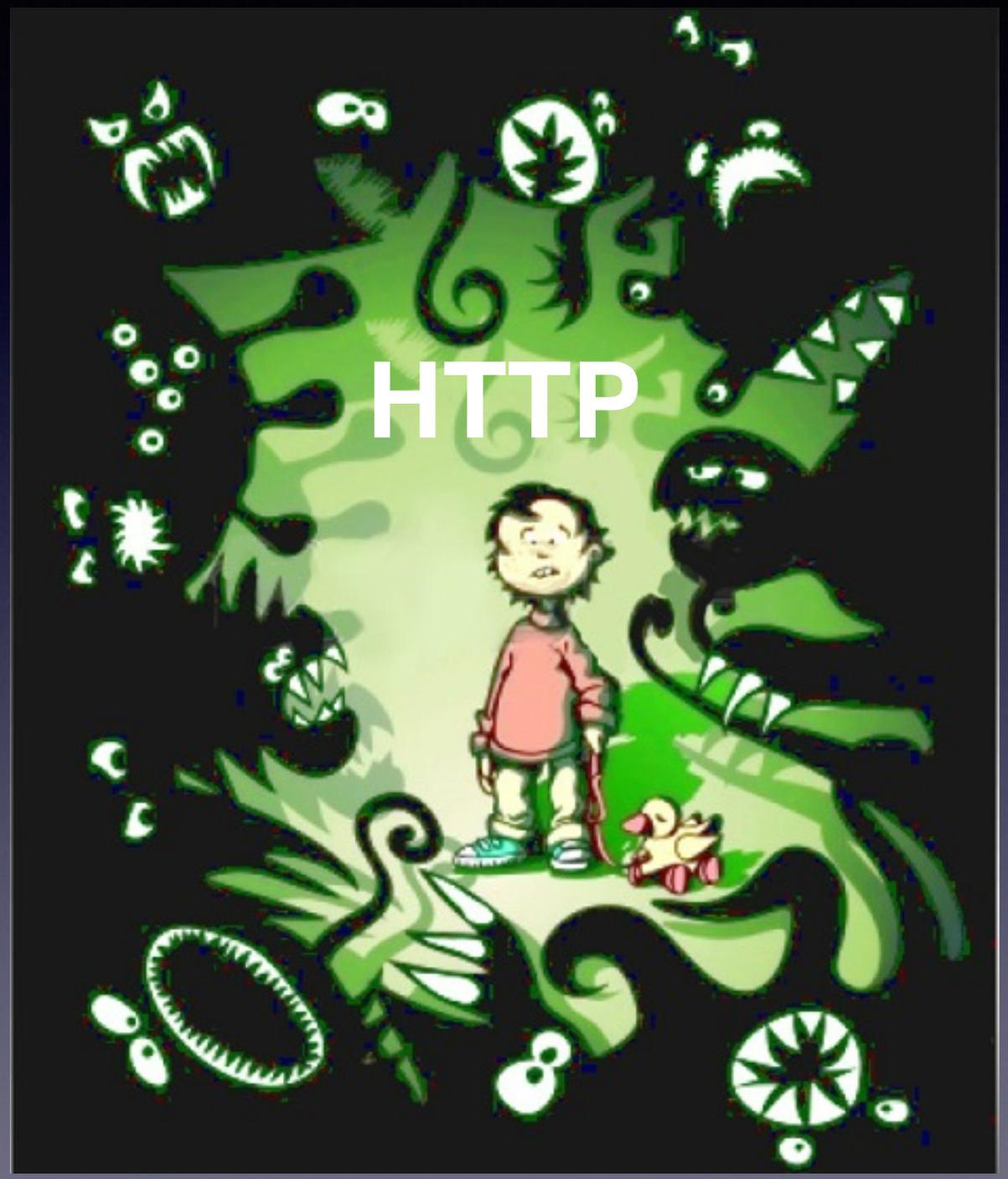

```
spawn_link(
 ?MODULE,
 send_s3_request,
 [S3url,AudioBin,self()]).
```

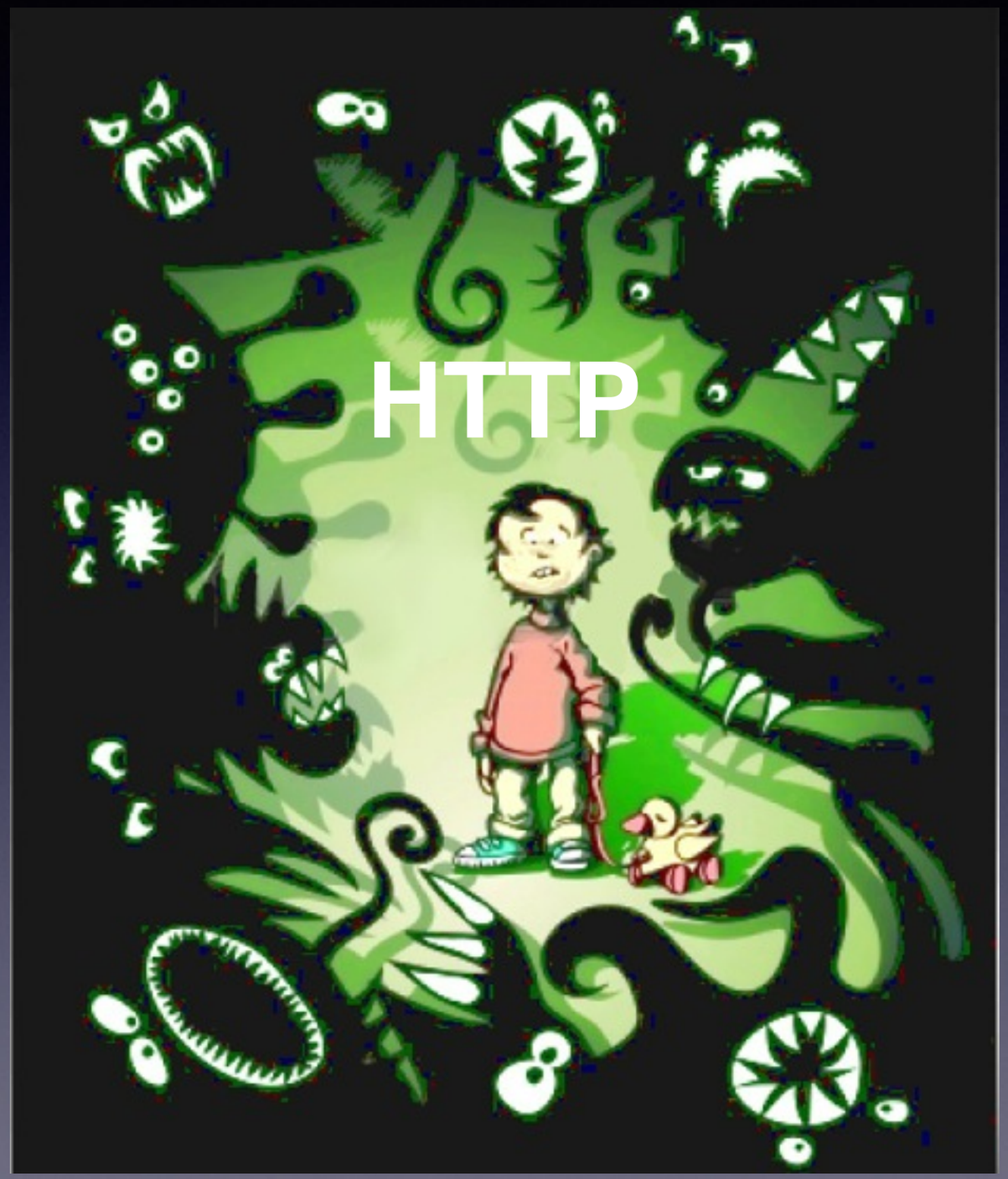
```
send_s3_request(S3url, AudioBin, Fsm)->
  S3Result = http_put(S3Url, AudioBin),
```

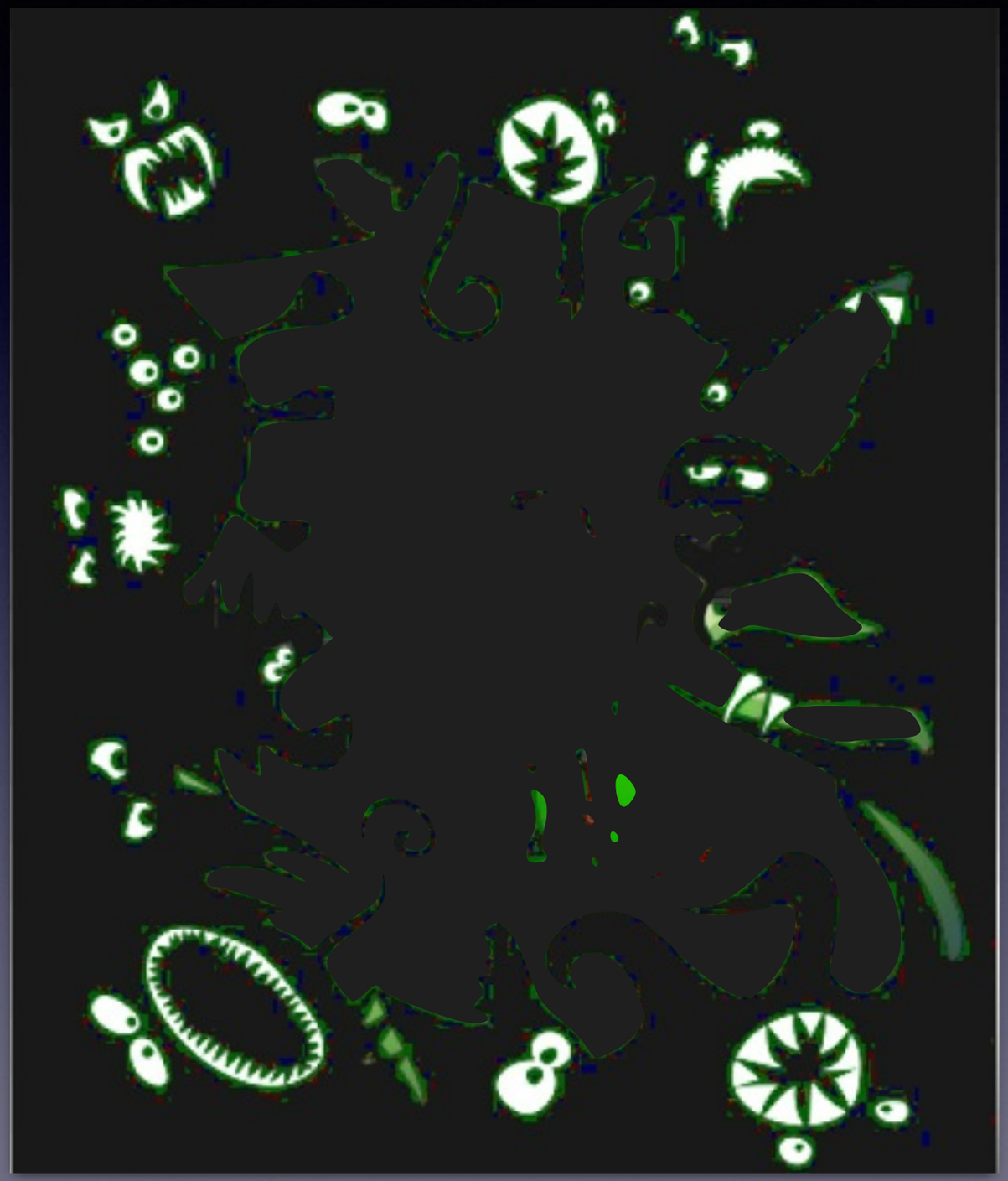

```
spawn_link(
  ?MODULE,
  send_s3_request,
  [S3url,AudioBin,self()]).
```

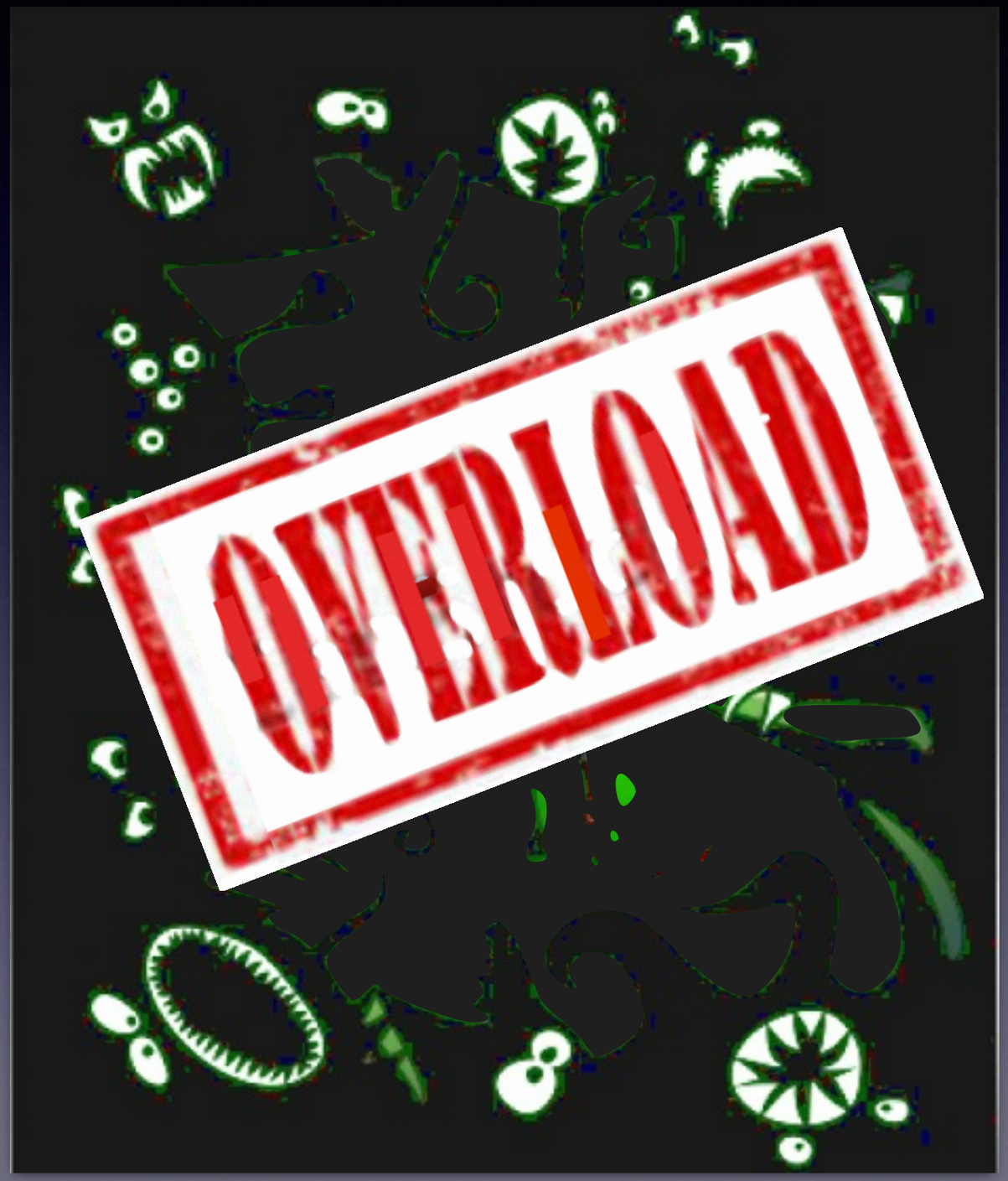
```
send_s3_request(S3url, AudioBin, Fsm)->
   S3Result = http_put(S3Url, AudioBin),
   send_result(Fsm, S3Result).
```


```
spawn_link(
 ?MODULE,
 send_s3_request,
 [S3url,AudioBin,self()]).
```


HTTP


HTTP



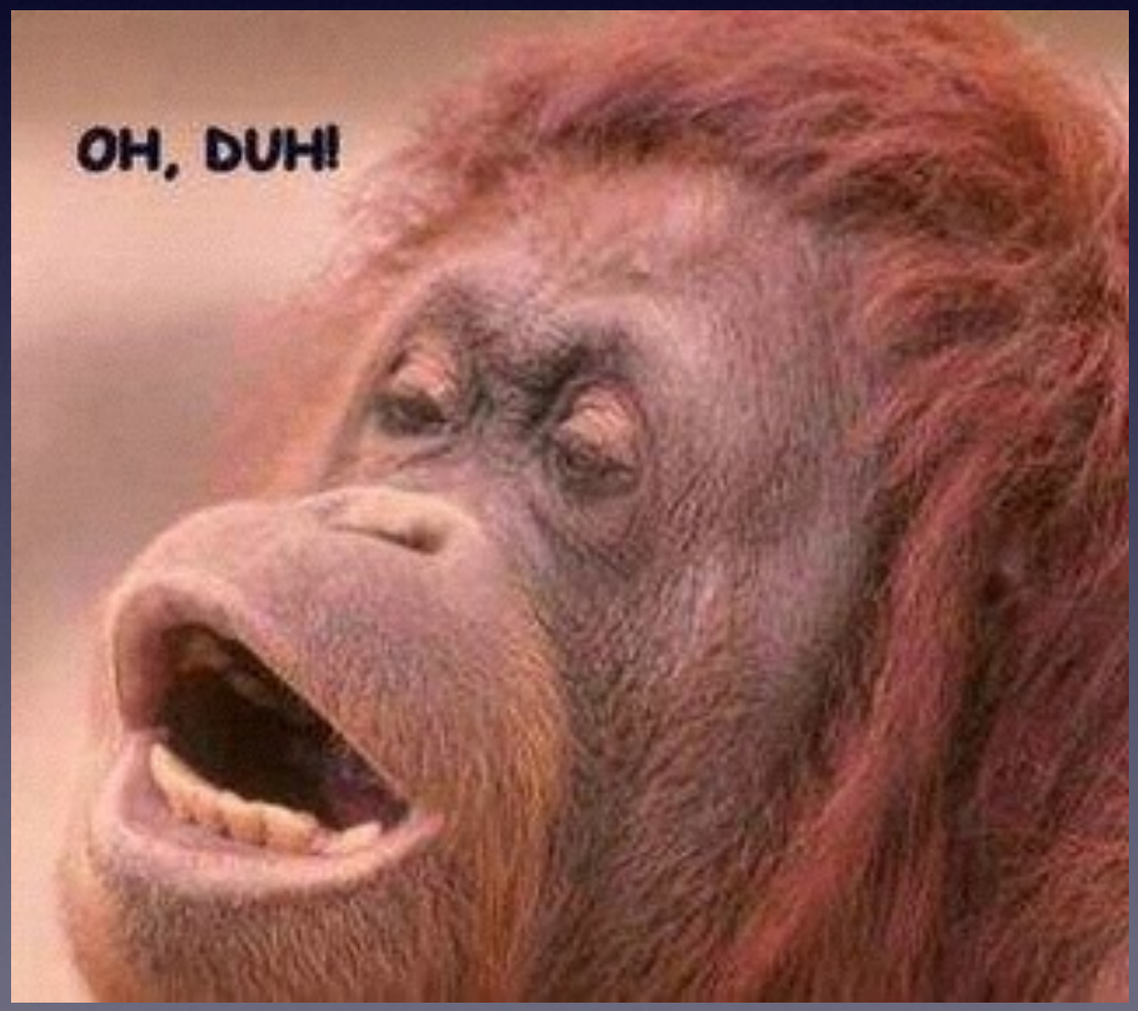

OVERLOAD

Limit simultaneous audio-conversion jobs to #CPUs - 1.

Limit simultaneous audio-conversion jobs to #CPUs - 1.

Reserve one CPU for high priority work.

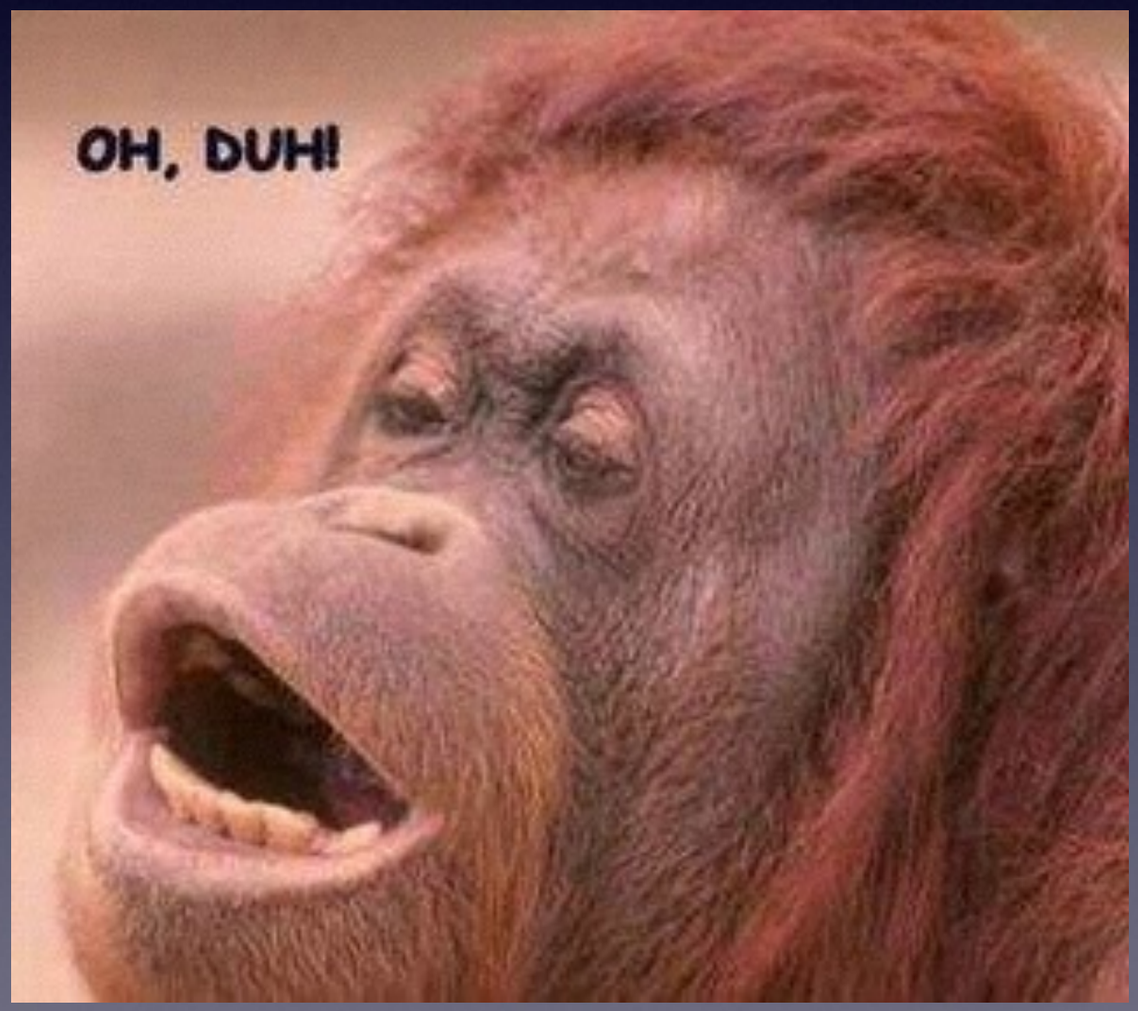
Limit simultaneous audio-conversion jobs to #CPUs - 1.



Reserve one CPU for high priority work.

Limit simultaneous audio-conversion jobs to #CPUs - 1.

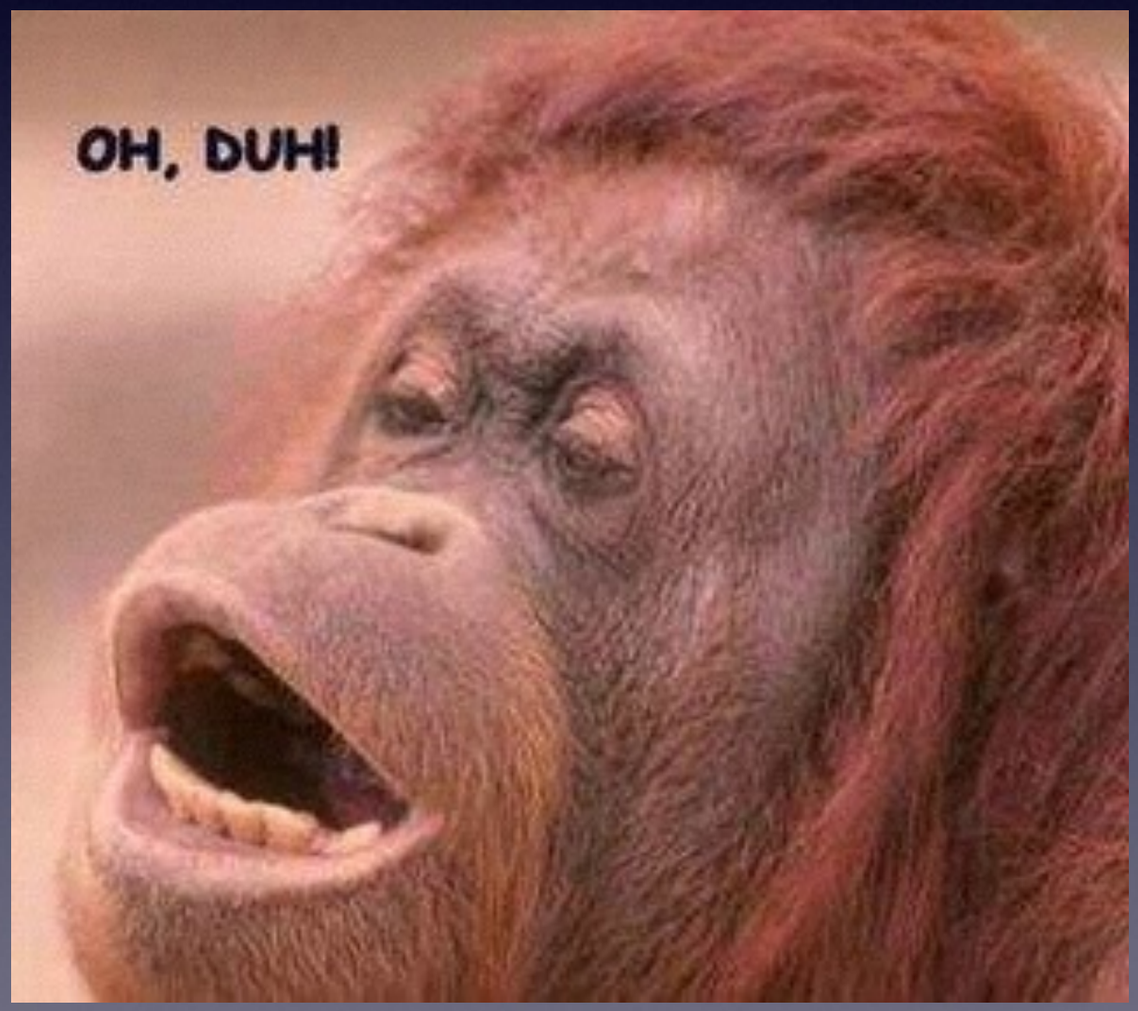
- Roll your own?



Reserve one CPU for high priority work.

Limit simultaneous audio-conversion jobs to #CPUs - 1.

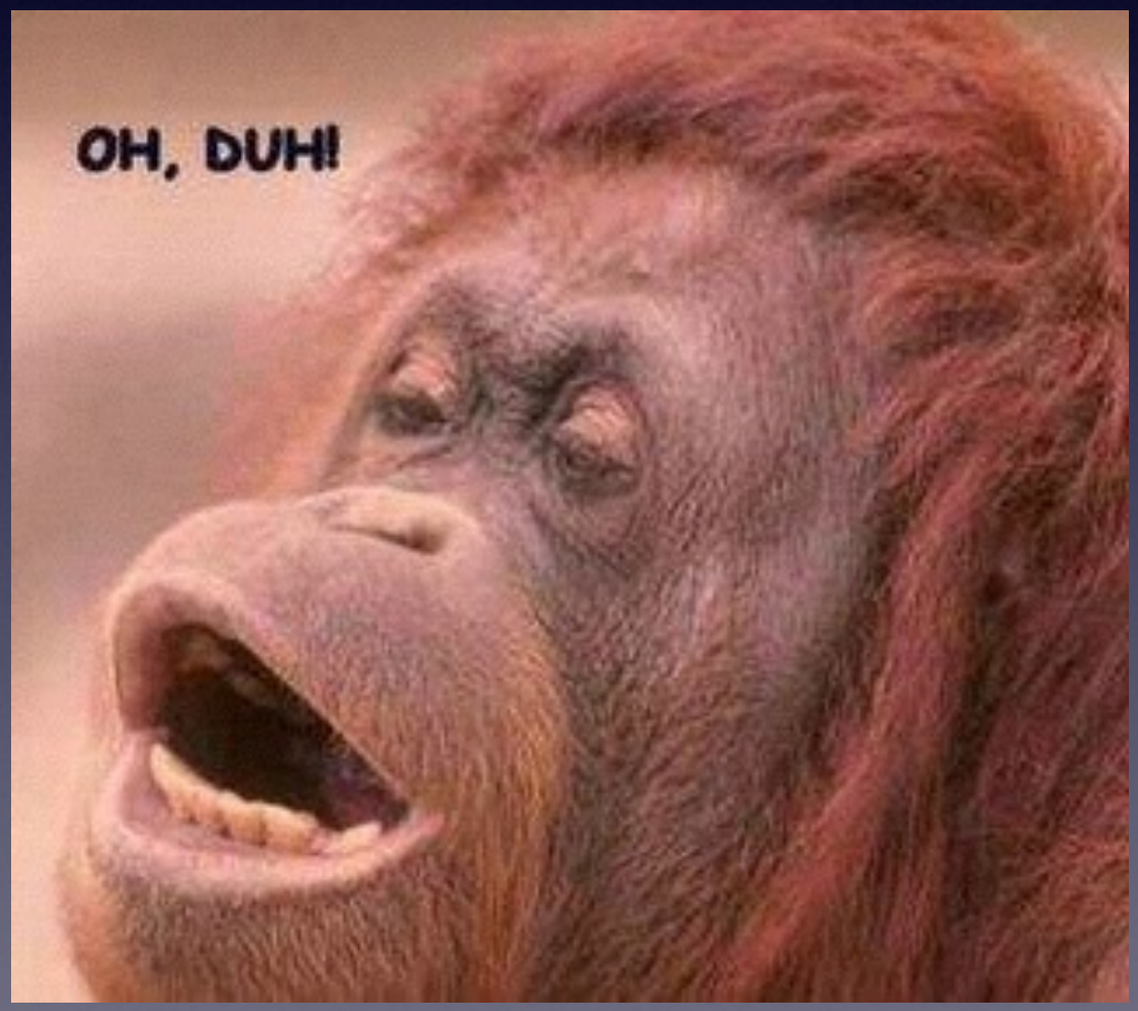
- Roll your own?
- Use existing library?



Reserve one CPU for high priority work.

Limit simultaneous audio-conversion jobs to #CPUs - 1.

- Roll your own?
- Use existing library?



Reserve one CPU for high priority work.

The **Jobs** Framework

by Ulf Wiger

The **Jobs** Framework by Ulf Wiger

“Job scheduler for load regulation” — Ulf Wiger

The **Jobs** Framework by Ulf Wiger

“Job scheduler for load regulation” — Ulf Wiger

Generic Load Regulation Framework for Erlang

August 5, 2010

Ulf Wiger

Erlang Solutions Ltd

ulf.wiger@erlang-solutions.com

lecoms, the domain for which Erlang was conceived, ıd ubiquitous requirements on overload protection, the platform offers no unified approach to addressing the e Erlang community mailing list frequently sports dis- how to avoid overload situations in individual com- processes, indicating that such an approach would be s Telecoms migrated from carefully regulated single- orks towards multimedia services on top of best-effort e packet data backbones, much was learned about pro- o-end quality of service with a network of loosely ıponents, with only basic means of prioritization and . This paper explores the similarity of such networks Erlang-based message-passing architectures, and ar- obust way of managing high-load conditions is to reg- ıput edges of the system, and sampling known internal in order to dynamically maintain optimum through- ion of typical overload conditions are discussed, and egulation framework – JOBS – is presented, together es of how such overload conditions can be mitigated.

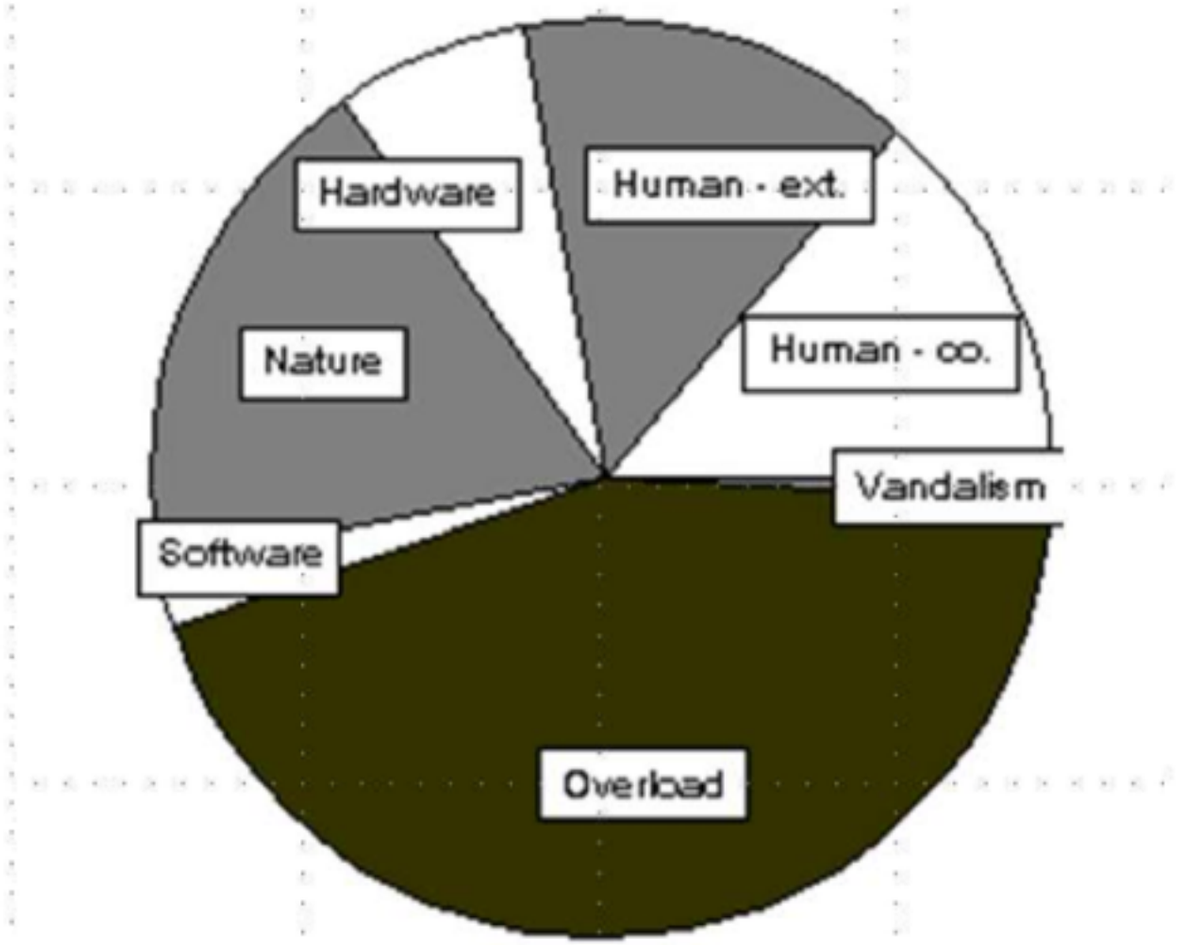


Figure 1. Failures in the US Public Switched Telephony Network, outage minutes by cause, see (Kuhn 1997).

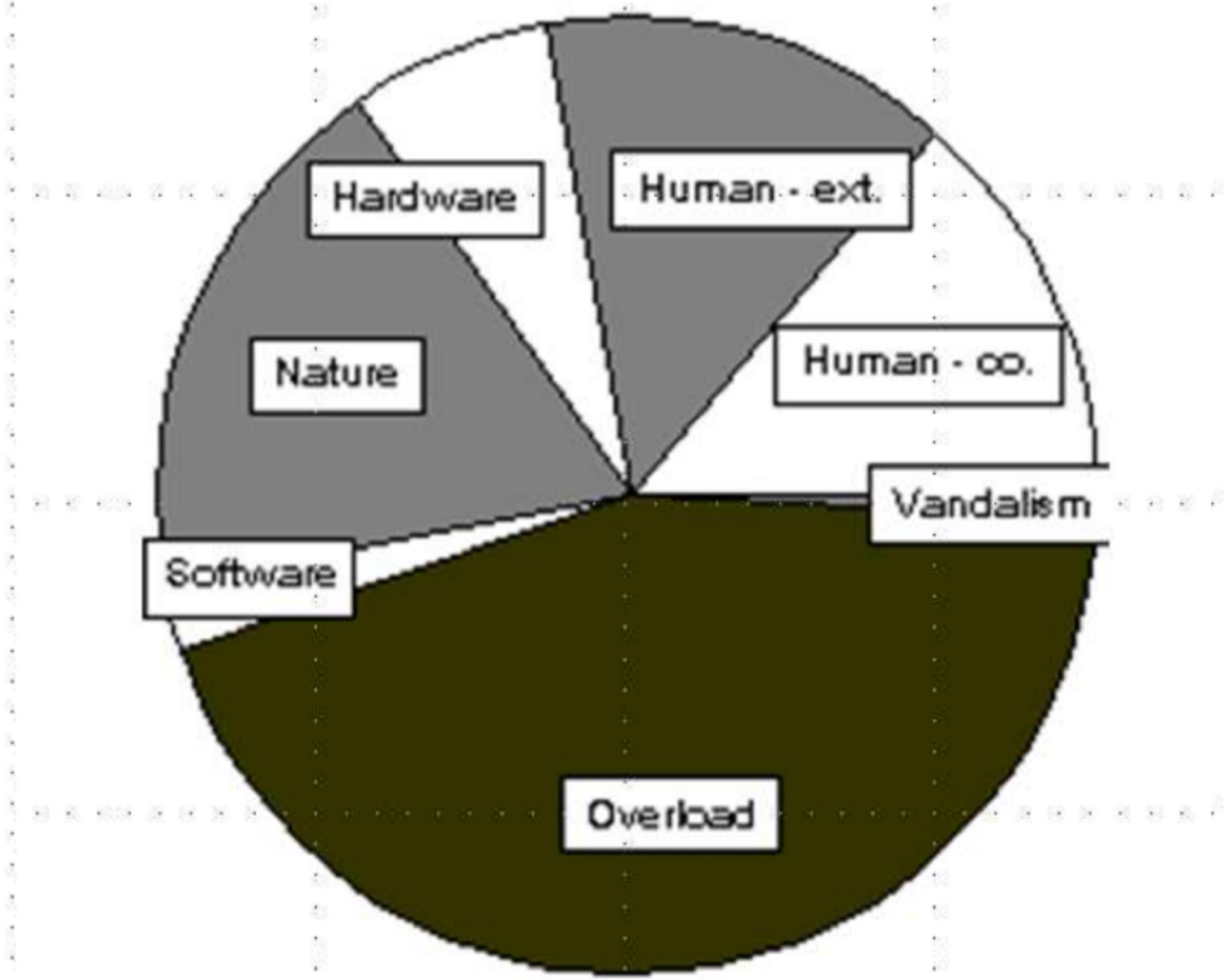


Figure 1. Failures in the US Public Switched Telephony Network, outage minutes by cause, see (Kuhn 1997).


1 2 3 4 5 6 7


1 2 3 4 5 6 7


b a
1 2 3 4 5 6 7


based
1 2 3 4 5 6 7


C
based
1 2 3 4 5 6 7


C
ased
1 2 3 4 5 6 7

Co[redacted]ased

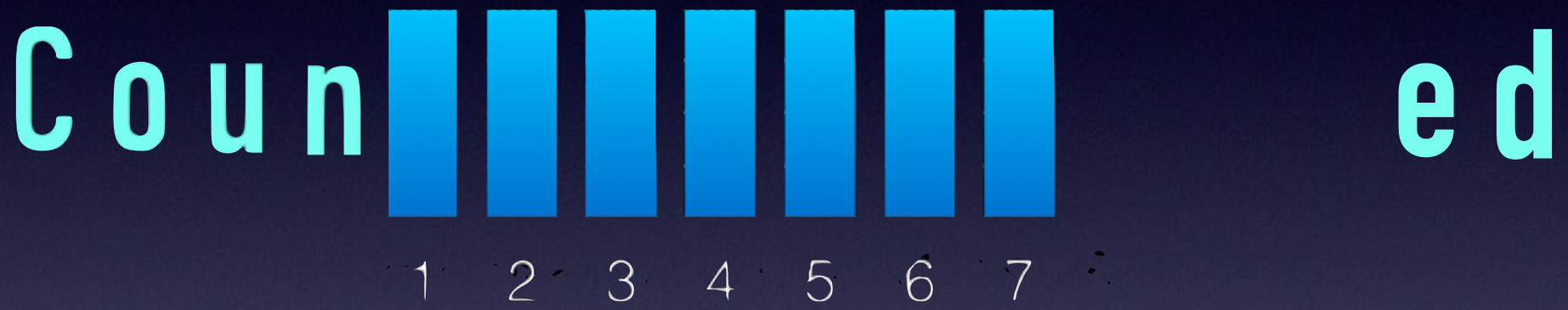
1 2 3 4 5 6 7

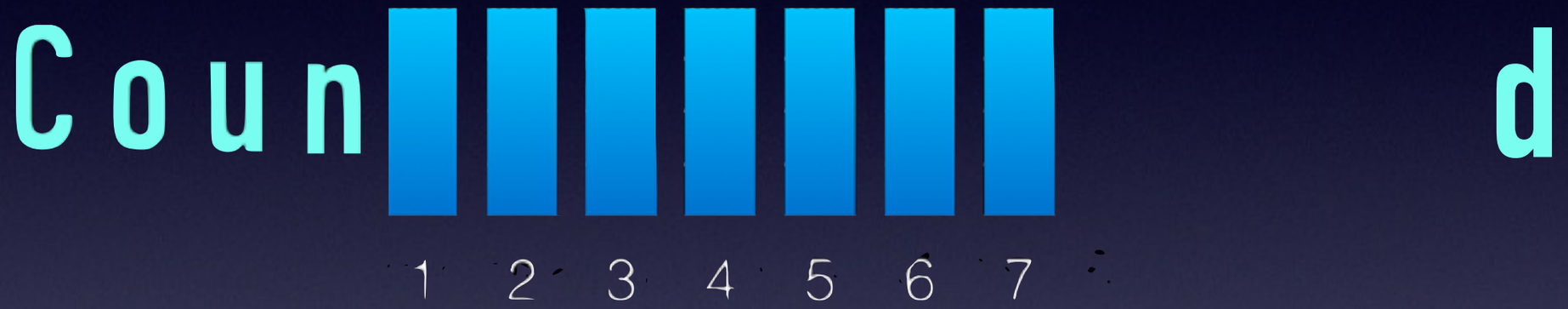
Cou[redacted]sed

1 2 3 4 5 6 7

Cou[redacted]ed

1 2 3 4 5 6 7


Coun
ed
1 2 3 4 5 6 7


Coun
d
1 2 3 4 5 6 7


Count
d
1 2 3 4 5 6 7

Count

1 2 3 4 5 6 7

Counter

1 2 3 4 5 6 7

1 2 3 4 5 6 7


Counter gula
1 2 3 4 5 6 7

Counter gulation

1 2 3 4 5 6 7

Counterbased Regulation

1 2 3 4 5 6 7

1. add “jobs” to **rebar.config** deps

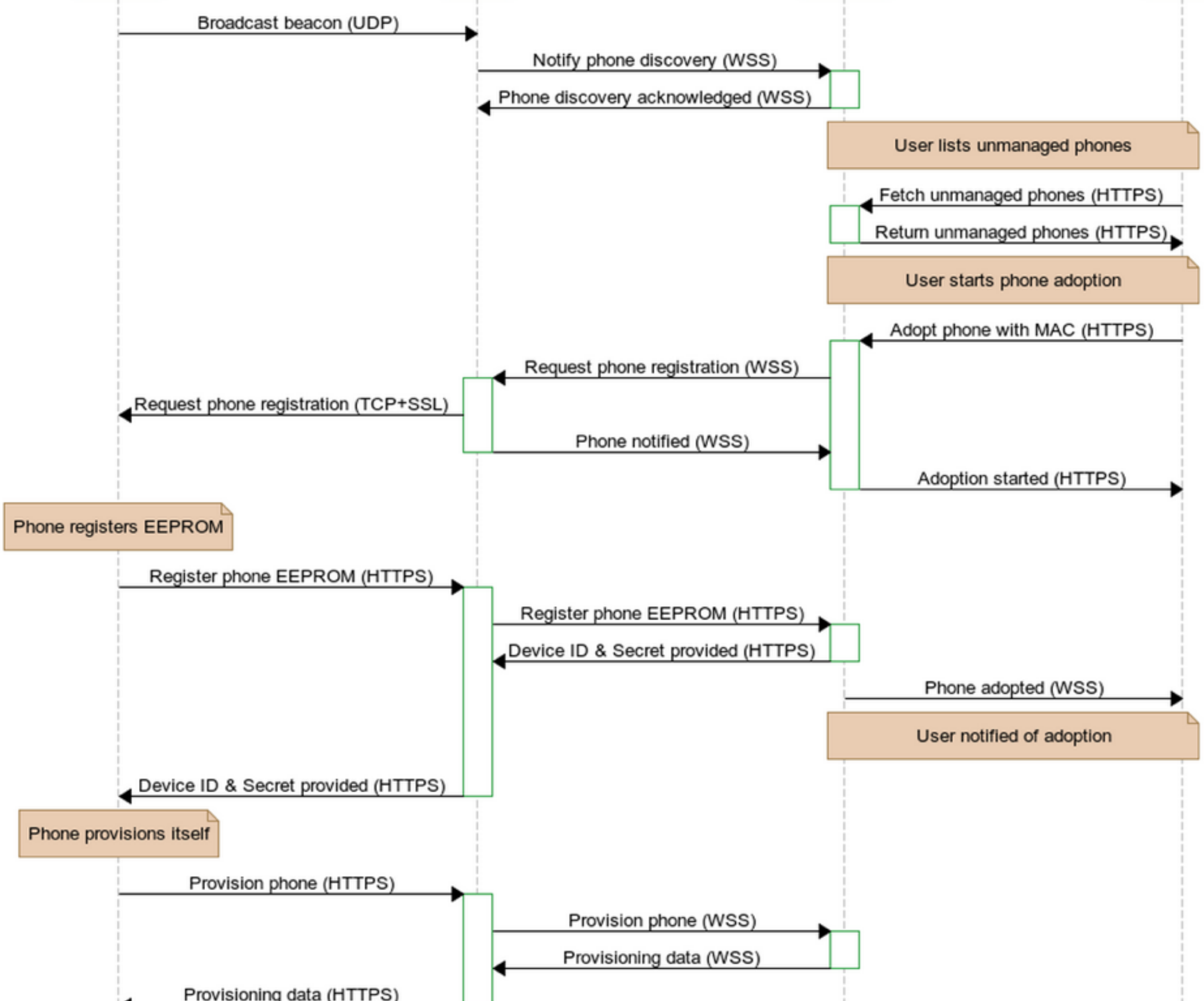
1. add “jobs” to **rebar.config** deps

```
{jobs, ".*", {git,
"https://github.com/uwiger/jobs",
{branch, “master"}}}
```

2. setup jobs env in **app.config**

2. setup jobs env in **app.config**

```
{jobs,[
        {queues, [
            {audio_conversions,[
                {regulators, [{counter, [{limit,7}]}]}
            ]}
          ]}
        ]
}
```


Broadcast beacon (UDP)
Notify phone discovery (WSS)
Phone discovery acknowledged (WSS)
User lists unmanaged phones
Fetch unmanaged phones (HTTPS)
Return unmanaged phones (HTTPS)
User starts phone adoption
Adopt phone with MAC (HTTPS)
Request phone registration (WSS)
Request phone registration (TCP+SSL)
Phone notified (WSS)
Adoption started (HTTPS)
Phone registers EEPROM
Register phone EEPROM (HTTPS)
Register phone EEPROM (HTTPS)
Device ID & Secret provided (HTTPS)
Phone adopted (WSS)
User notified of adoption
Device ID & Secret provided (HTTPS)
Phone provisions itself
Provision phone (HTTPS)
Provision phone (WSS)
Provisioning data (WSS)
Provisioning data (HTTPS)

```
jobs:run(audio_conversions,fun()->
do_audio_conversion() end)
```

```
 ok;
dle_ac({wav, ItemKeyUrl}, Rest, From, State)->
 spawn_ac_handler(Rest ++ [{wav, ItemKeyUrl}], From, State);
dle_ac({Format, ItemKeyUrl}, Rest, From, State)->
 handle_output_format(Format, ItemKeyUrl, From, State),
 spawn_ac_handler(Rest, From, State).

e_name(ItemKeyUrl)->
 string:join(string:tokens(ItemKeyUrl, "/"), "_").

(Text, Lang, VoiceActor, ItemKeyUrl, State)->
 jobs:run(ac, fun()-> do_tts(Text, Lang, VoiceActor, ItemKeyUrl, State) end).
tts(Text, Lang, VoiceActor, ItemKeyUrl, #state{scratch_dir = ScratchDir} = State) ->
 FilePath = file_name(ItemKeyUrl),
 TextFilePath = temp_text_file(Text, ScratchDir),
 WavFilePath = ScratchDir ++ FilePath ++ ".wav",
 TtsCommand = tts_command(TextFilePath, WavFilePath,
     actor(Lang,VoiceActor,State),
     State),
 Result =
     case command(TtsCommand) of
         ok -> {ok, filelib:file_size(WavFilePath)};
         {error, Error}-> {error, Error}
     end,
 file:delete(TextFilePath),
 Result.

dle_output_format(Format, ItemKeyUrl, From, #state{scratch_dir = ScratchDir} = State)->
 FileName = ScratchDir ++ file_name(ItemKeyUrl),
 FileNameWithExt = FileName ++ audio_file_ext(Format),
 case filelib:file_size(FileNameWithExt) > 0 of
     false ->
         convert(wav, Format, FileName, State);
     true -> ok
 end,
 save(Format, FileNameWithExt, ItemKeyUrl, From, State).

vert(FormatFrom, FormatTo, FileName, State)->
 jobs:run(ac, fun() -> do_convert(FormatFrom, FormatTo, FileName, State) end).

convert(FormatFrom, FormatTo, FileName, #state{exec_path = ExecPath})->
 command(ExecPath ++ conversion_command(FormatFrom, FormatTo,
```

```
 ok;
dle_ac({wav, ItemKeyUrl}, Rest, From, State)->
 spawn_ac_handler(Rest ++ [{wav, ItemKeyUrl}], From, State);
dle_ac({Format, ItemKeyUrl}, Rest, From, State)->
 handle_output_format(Format, ItemKeyUrl, From, State),
 spawn_ac_handler(Rest, From, State).

e_name(ItemKeyUrl)->
 string:join(string:tokens(ItemKeyUrl, "/"), "_").

tts(Text, Lang, VoiceActor, ItemKeyUrl, State)->
    jobs:run(ac, fun()-> do_tts(Text, Lang, VoiceActor, ItemKeyUrl, State)
 FilePath = file_name(ItemKeyUrl),
 TextFilePath = temp_text_file(Text, ScratchDir),
 WavFilePath = ScratchDir ++ FilePath ++ ".wav",
 TtsCommand = tts_command(TextFilePath, WavFilePath,
     actor(Lang,VoiceActor,State),
     State),
 Result =
     case command(TtsCommand) of
         ok -> {ok, filelib:file_size(WavFilePath)};
         {error, Error}-> {error, Error}
     end,
 file:delete(TextFilePath),
 Result.

dle_output_format(Format, ItemKeyUrl, From, #state{scratch_dir = ScratchDir} = State)->
 FileName = ScratchDir ++ file_name(ItemKeyUrl),
 FileNameWithExt = FileName ++ audio_file_ext(Format),
 case filelib:file_size(FileNameWithExt) > 0 of
     false ->
         convert(wav, Format, FileName, State);
     true -> ok
 end,
 save(Format, FileNameWithExt, ItemKeyUrl, From, State).

vert(FormatFrom, FormatTo, FileName, State)->
 jobs:run(ac, fun() -> do_convert(FormatFrom, FormatTo, FileName, State) end).

convert(FormatFrom, FormatTo, FileName, #state{exec_path = ExecPath})->
 command(ExecPath ++ conversion_command(FormatFrom, FormatTo,
```

ac_server.app.src

```
{application, ac_server,
 [
  {description, “My Audio Conversion Application"},
  {vsn, "0.0.1"},
  {registered, []},
  {applications, [
                  kernel,
                  stdlib,
                  crypto,
                  lager,
                  proper,
                  reltool_util,
                  os_mon,
                  jobs
                 ]},
  {mod, { ac_server_app, []}},
  {env, []}
 ]}.
```

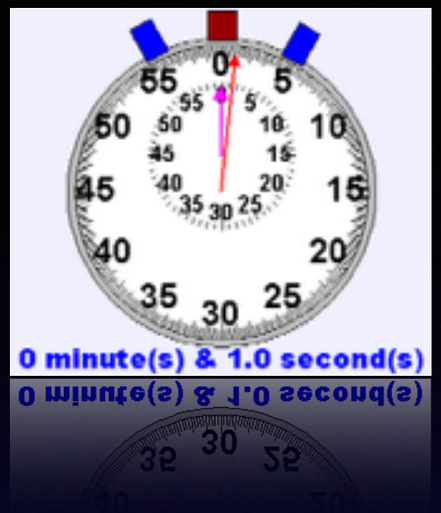

KITCHEN


KITCHEN

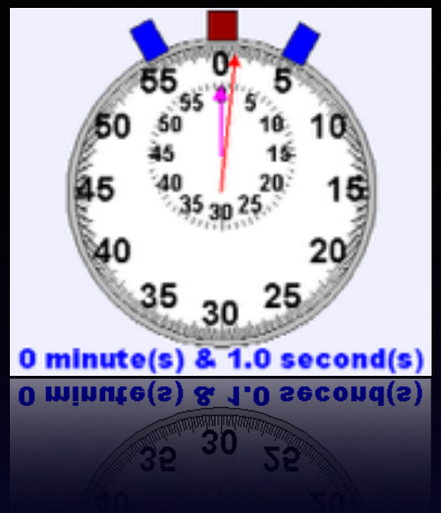

0 minute(s) & 1.0 second(s)


0 minute(s) & 1.0 second(s)


0 minute(s) & 1.0 second(s)



Given a frequency *f*, ensures that the rate of accepted jobs does not exceed *f*



Given a frequency *f*, ensures that the rate of accepted jobs does not exceed *f*

```
{jobs,[
        {queues, [
            {http_requests,[
                {regulators, [{rate, [{limit,1000}]}]}
            ]}
          ]}
        ]
}
```

What if…

What if...

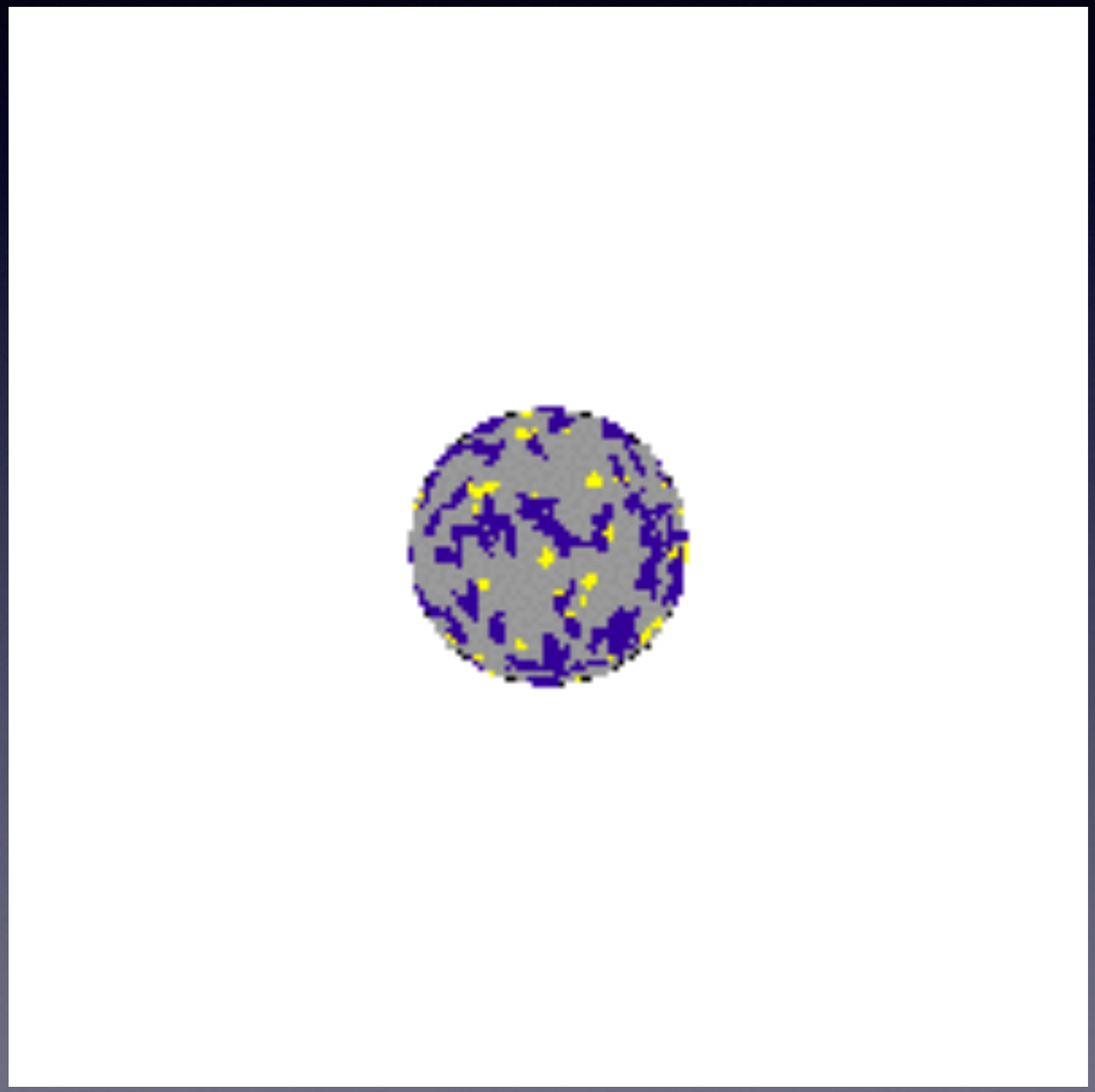
What if…

…we get a denial of service attack?

What if...

What if...

What if...



HTTP_REQUESTS QUEUE

What if...

Limit the queue size!

Limit the queue size!

```
{jobs,[
        {queues, [
            {http_requests,[
                {max_size, 5000},
                {regulators, [{rate, [{limit,1000}]}]}]}
            ]}
        ]
}
```

What if...

What if...

Our requests have a strict timeout?

Limit the request wait time in the queue!

Limit the request wait time in the queue!

```
{jobs,[{queues, [{rtb_requests,
[{max_time, 200}, {regulators,…
```

Limit the request wait time in the queue!

```
{jobs,[{queues, [{rtb_requests,
[{max_time, 200}, {regulators,…
```

```
{jobs,[{queues, [{payment_requests,
[{max_time, 600000}, {regulators,…
```


0 minute(s) & 1.0 second(s)


0 minute(s) & 1.0 second(s)


0 minute(s) & 1.0 second(s)

The group-rate regulator lets us put a cap on how many jobs from a **group** of queues can execute per second

```
{jobs,
 [{group_rates,[
   {http_request_rate, [{limit,1000}]}
  ]},
```

```
{jobs,
 [{group_rates,[
   {http_request_rate, [{limit,1000}]}
  ]},
   {queues, [
     {rtb_requests,
       [{max_time, 200},
        {regulators,[{group_rate,http_request_rate}]}
       ]
     },
```

```
{jobs,
 [{group_rates,[
   {http_request_rate, [{limit,1000}]}
  ]},
   {queues, [
     {rtb_requests,
        [{max_time, 200},
         {regulators,[{group_rate,http_request_rate}]}
        ]
     },
     {payment_requests,
        [{max_time, 600000},
         {regulators,[{group_rate,http_request_rate}]}
        ]
     }]
    }
  ]
}
```

What if...

What if...

#CPUs - 1 is not good enough?

Sampler Framework

Sampler Framework

sends feedback indicators: cpu, memory…

Sampler Framework

sends feedback indicators: cpu, memory…

LOAD FACTORS FROM
LOCAL + REMOTE NODES =
% BY WHICH TO REDUCE
PREDEFINED REGULATOR
LIMIT

Sampler Framework

sends feedback indicators: cpu, memory…

LOAD FACTORS FROM
REMOTE NODES?

LOAD FACTORS FROM
LOCAL + REMOTE NODES =
% BY WHICH TO REDUCE
PREDEFINED REGULATOR
LIMIT

Sampler Framework

sends feedback indicators: cpu, memory…

LOAD FACTORS FROM LOCAL + REMOTE NODES = % BY WHICH TO REDUCE PREDEFINED REGULATOR LIMIT

LOAD FACTORS FROM REMOTE NODES?

```
{jobs,[
  {samplers,[
    {cpu,jobs_sampler_cpu,[]}
  ]}
  {queues, [
     {http_requests,[
       {regulators,
         [{rate,[
            {limit,1000},
            {modifiers,[{    ,10}]}
          ]}
         ]
     ]}
  ]}
]}
```

```
{jobs,[
  {samplers,[
    {cpu,jobs_sampler_cpu,[]}
  ]}
  {queues, [
     {http_requests,[
       {regulators,
         [{rate,[
            {limit,1000},
            {modifiers,[{cpu,10}]}
          ]}
         ]
     ]}
  ]}
]}
```

```
{jobs,[
  {samplers,[
    {cpu,jobs_sampler_cpu,[]}
  ]}
  {queues, [
     {http_requests,[
       {regulators,
         [{rate,[
            {limit,1000},
            {modifiers,[{cpu,10}]}
          ]}
         ]
     ]}
  ]}
]}
```

jobs_sampler_cpu

jobs_sampler_cpu

- Uses **os_mon cpu_sup***

---

*CPU Load and CPU Utilization Supervisor Process

jobs_sampler_cpu

- Uses **os_mon cpu_sup***

cpu_sup:util([per_cpu])

---

*CPU Load and CPU Utilization Supervisor Process

jobs_sampler_cpu

- Uses **os_mon cpu_sup***

---

*CPU Load and CPU Utilization Supervisor Process

jobs_sampler_cpu

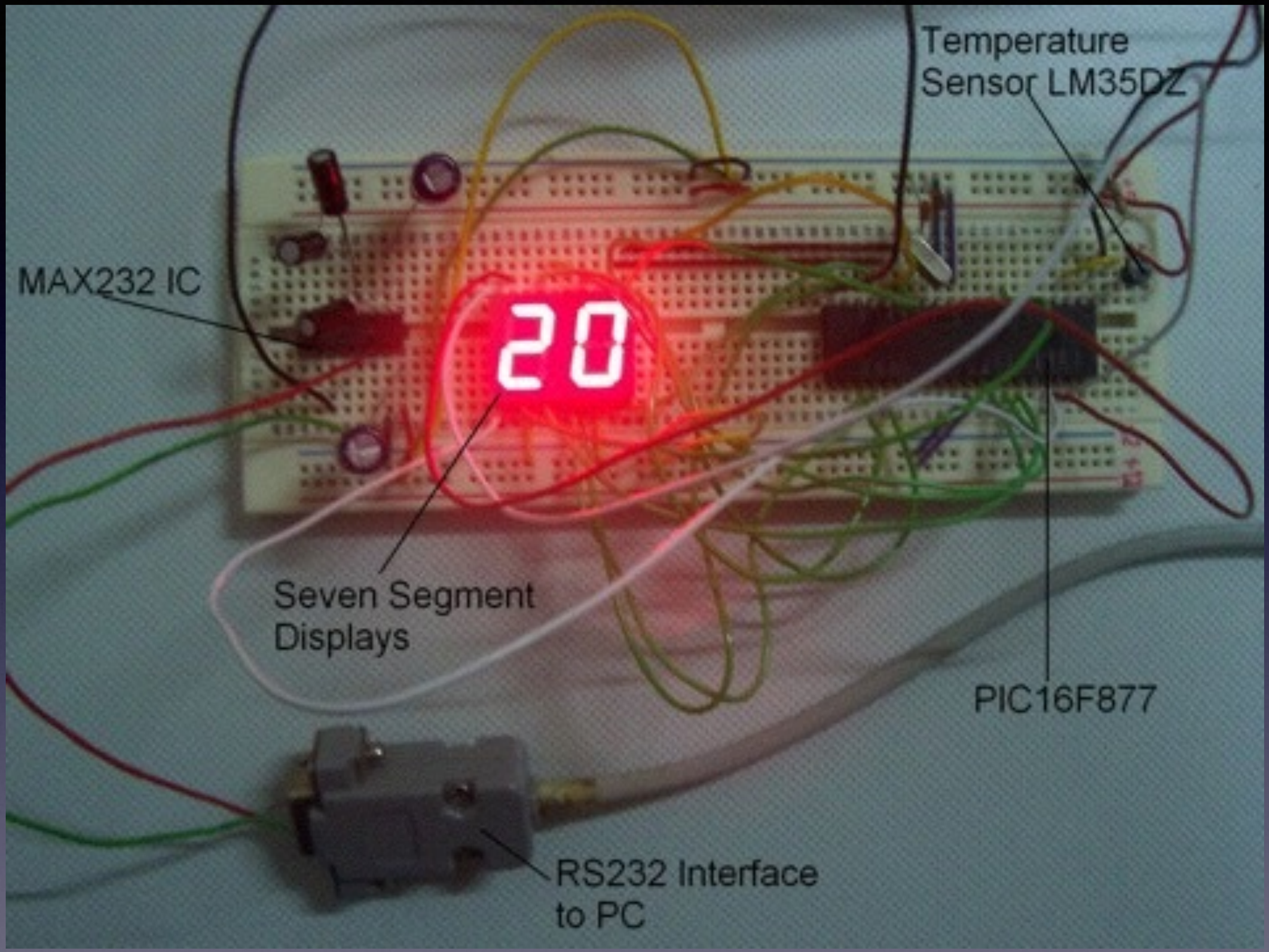
- Uses **os_mon cpu_sup***
- Available only on Unix/Linux OS

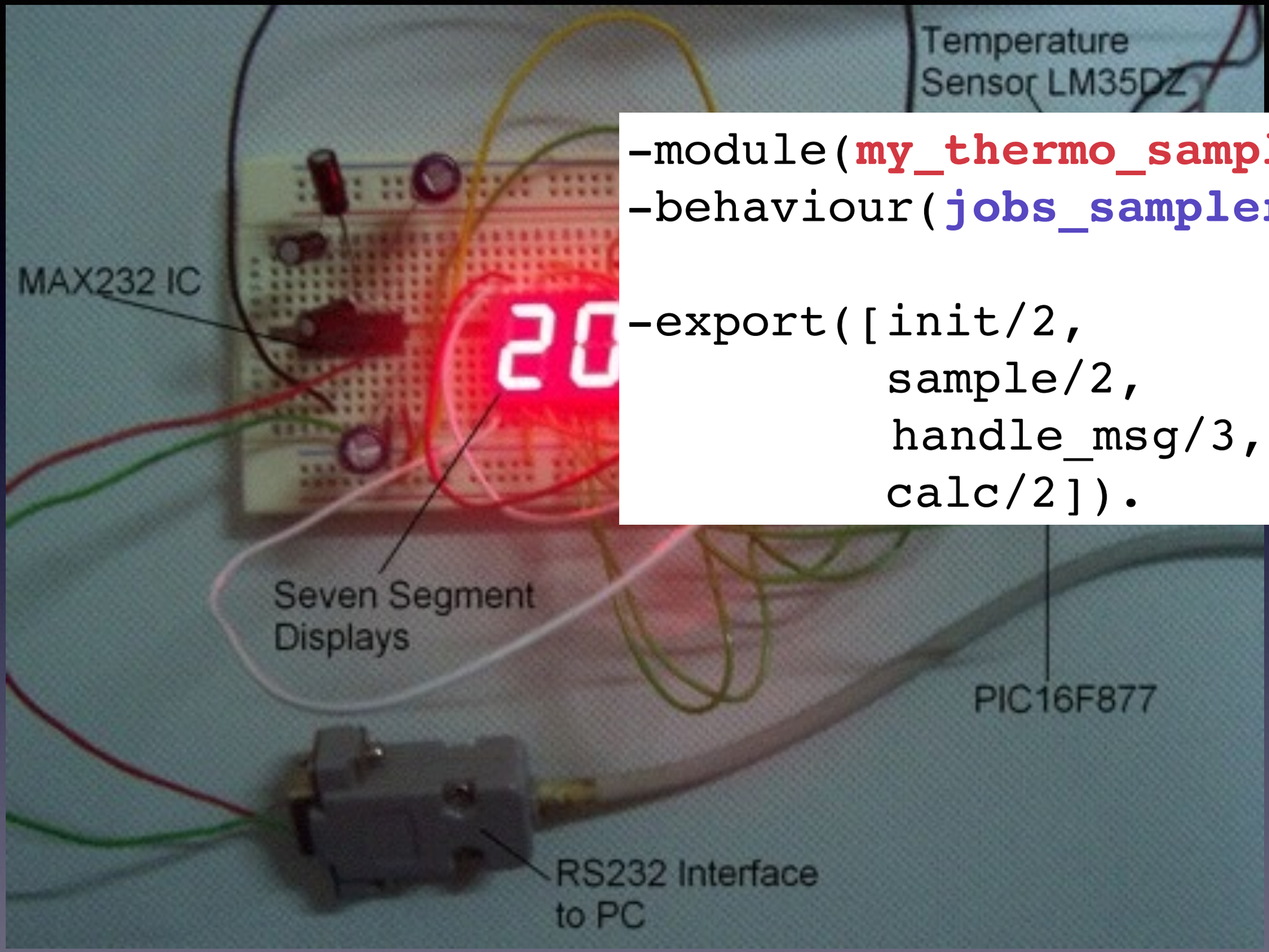
---

*CPU Load and CPU Utilization Supervisor Process

Write your own sampler plug-ins…

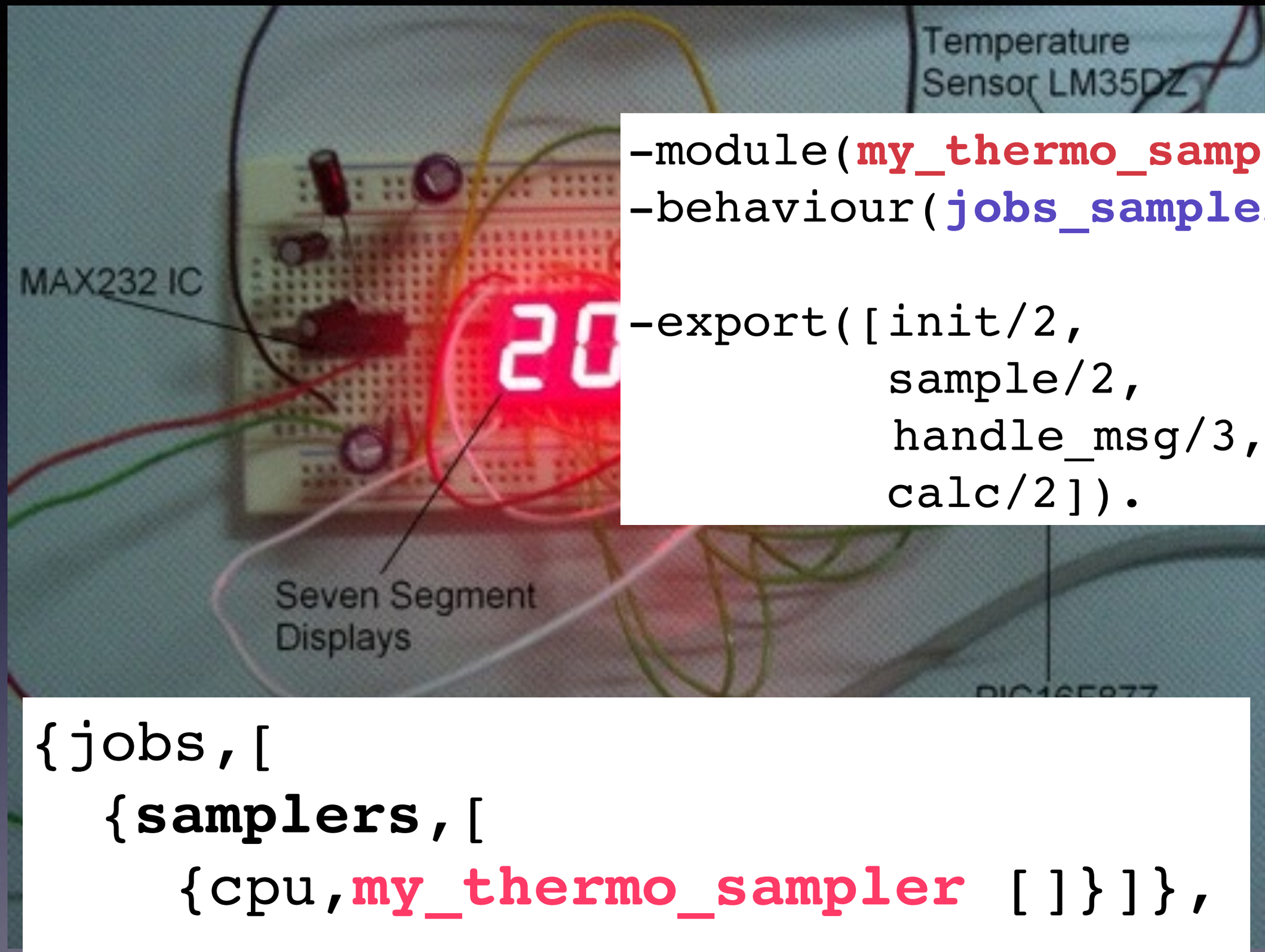
Write your own sampler plug-ins…


Temperature
Sensor LM35DZ
MAX232 IC
20
Seven Segment
Displays
PIC16F877
RS232 Interface
to PC




```
-module(my_thermo_sampler).
-behaviour(jobs_sampler).

-export([init/2,
         sample/2,
         handle_msg/3,
         calc/2]).
```




```
-module(my_thermo_sampler).
-behaviour(jobs_sampler).

-export([init/2,
         sample/2,
         handle_msg/3,
         calc/2]).
```

```
{jobs,[
  {samplers,[
    {cpu,my_thermo_sampler [ ]}]},
…
```

What if...

What if…

POST requests with varying size files get queued up

memory_friendly_queue

memory_friendly_queue

- max_size in bytes instead of number of entries

memory_friendly_queue

- max_size in bytes instead of number of entries
- when max_size is reached start swapping queue entries to a disk partition

```
-module(memory_friendly_queue).
-behaviour(jobs_queue).

-export([new/2,
         delete/1,
         in/3,
         out/2,
         peek/1,
         info/2,
         all/1]).
```

```
-module(memory_friendly_queue).
-behaviour(jobs_queue).

-export([new/2,
         delete/1,
         in/3,
         out/2,
         peek/1,
         info/2,
         all/1]).
```

```
{jobs,[
        {queues, [
            {http_requests,[
                {mod, memory_friendly_queue},
                {max_size, 100000} %now in bytes
                {regulators, [{rate, [{limit,1000}]}]}]}
            ]}
          ]}
        ]
```

free_RAM_sampler

free_RAM_sampler

- Use **os_mon** **memsup** process.

```
-module(free_RAM_sampler).
-behaviour(jobs_sampler).

-export([init/2,
         sample/2,
         handle_msg/3,
         calc/2]).

sample(_Timestamp, State)->
  [   {total_memory,_TM},
      {free_memory,FreeMem},
       {system_total_memory,_STM} ] =
  memsup:get_system_memory_data(),
   %% Use FreeMem to compute load factor
```

```
-module(free_RAM_sampler).
-behaviour(jobs_sampler).

-export([init/2,
         sample/2,
         handle_msg/3,
         calc/2]).

sample(_Timestamp, State)->
[   {total_memory,_TM},
    {free_memory,FreeMem},
     {system_total_memory,_STM} ] =
 memsup:get_system_memory_data(),
  %% Use FreeMem to compute load factor
```

```
-module(free_RAM_sampler).
-behaviour(jobs_sampler).

-export([init/2,
         sample/2,
         handle_msg/3,
         calc/2]).

sample(_Timestamp, State)->
[   {total_memory,_TM},
    {free_memory,FreeMem},
     {system_total_memory,_STM} ] =
  memsup:get_system_memory_data(),
   %% Use FreeMem to compute load factor
```

Pattern matching

```
-module(free_RAM_sampler).
-behaviour(jobs_sampler).

-export([init/2,
         sample/2,
         handle_msg/3,
         calc/2]).

sample(_Timestamp, State)->
  [   {total_memory,_TM},
      {free_memory,FreeMem},
       {system_total_memory,_STM} ] =
  memsup:get_system_memory_data(),
   %% Use FreeMem to compute load factor
```

Adjust queue max_size.

Adjust queue max_size.

```
jobs:modify_queue(memory_friendly_queue,
[{max_size, 80000}]).
```

How to adjust queue max_size on the fly based on the info from our free_RAM_sampler?

How to adjust queue max_size on the fly based on the info from our free_RAM_sampler?

- Ask Ulf to add this functionality to the “jobs” server.

How to adjust queue max_size on the fly based on the info from our free_RAM_sampler?

- Ask Ulf to add this functionality to the “jobs” server.
- Help expand “jobs” functionality myself.

How to adjust queue max_size on the fly based on the info from our free_RAM_sampler?

- Ask Ulf to add this functionality to the “jobs” server.
- Help expand “jobs” functionality myself.
- Use the backdoor

- Call **jobs_sampler:subscribe()** in your own process

- Call **jobs_sampler:subscribe()** in your own process
- Now your process is getting feedback in its mailbox!

- Call **jobs_sampler:subscribe()** in your own process
- Now your process is getting feedback in its mailbox!
- Modify **memory_friendly_queue** **max_size** based on received sampler feedback:

- Call **jobs_sampler:subscribe()** in your own process
- Now your process is getting feedback in its mailbox!
- Modify **memory_friendly_queue** **max_size** based on received sampler feedback:

```
jobs:modify_queue(memory_friendly_queue,
[{max_size,
NEWLY_CALCULATED_SIZE_GOES_HERE}]).
```

True or false?

True or false?

Spawning processes in Erlang is very cheap.
Spawn all you want!

True or false?

- ✓ Spawning processes in Erlang is very cheap. Spawn all you want!

True or false?

- ✓ Spawning processes in Erlang is very cheap. Spawn all you want!

Don't worry a bit about what kind of work these processes are doing... Erlang VM will make sure everything just works!

True or false?

✓ Spawning processes in Erlang is very cheap.
Spawn all you want!

✗ Don't worry a bit about what kind of
work these processes are doing...
Erlang VM will make sure everything just works!

True or false?

✓ Spawning processes in Erlang is very cheap.
Spawn all you want!

✗ Don't worry a bit about what kind of
work these processes are doing...
Erlang VM will make sure everything just works!

Load regulation is trivial. Always consider
writing your own library.

True or false?

✓ Spawning processes in Erlang is very cheap. Spawn all you want!

✗ Don't worry a bit about what kind of work these processes are doing... Erlang VM will make sure everything just works!

✗ Load regulation is trivial. Always consider writing your own library.

True or false?

✓ Spawning processes in Erlang is very cheap. Spawn all you want!

✗ Don't worry a bit about what kind of work these processes are doing... Erlang VM will make sure everything just works!

✗ Load regulation is trivial. Always consider writing your own library.

Write CPU-intensive computations in C. Let Erlang manage concurrent execution.

True or false?

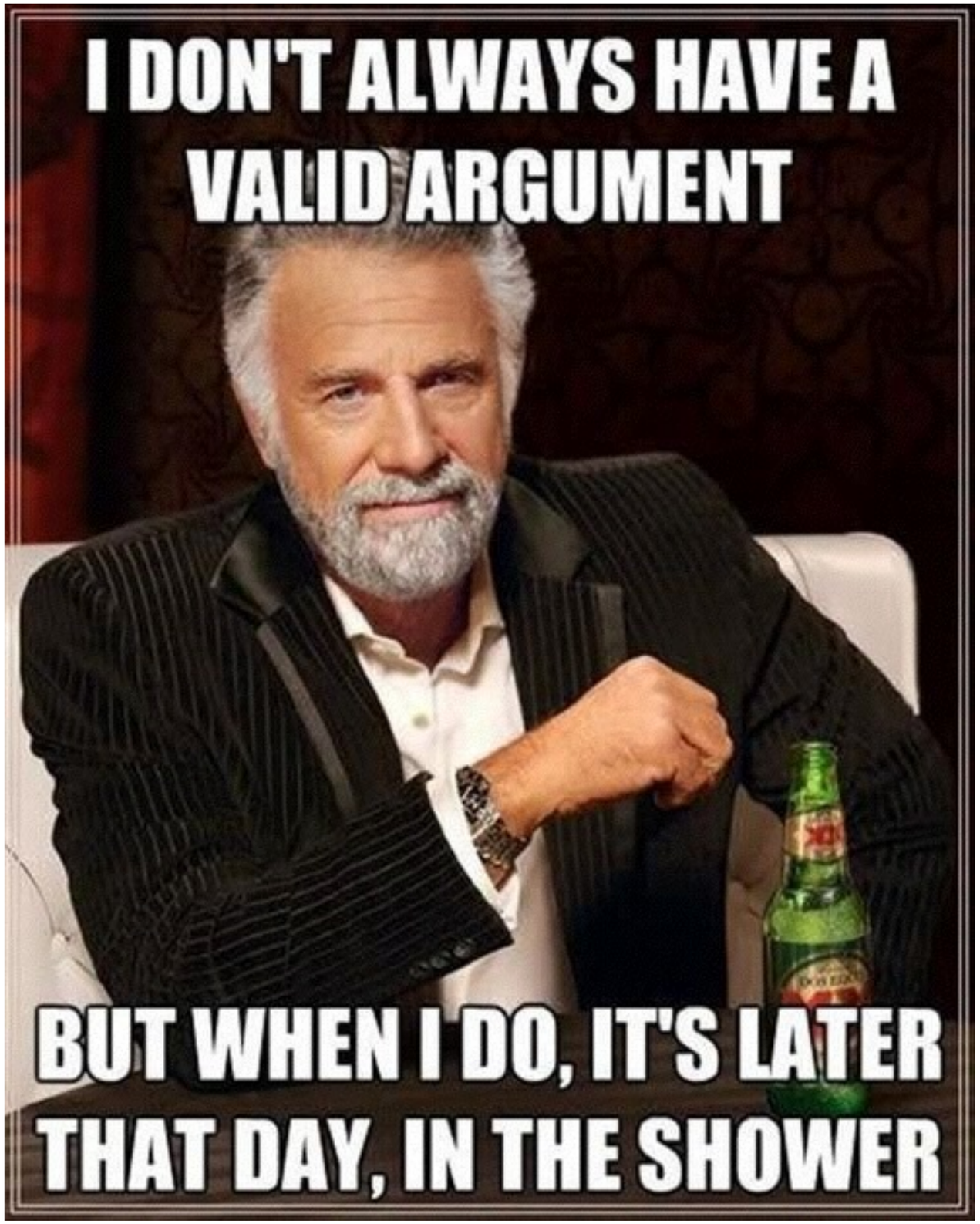
✓ Spawning processes in Erlang is very cheap. Spawn all you want!

✗ Don't worry a bit about what kind of work these processes are doing... Erlang VM will make sure everything just works!

✗ Load regulation is trivial. Always consider writing your own library.

✓ Write CPU-intensive computations in C. Let Erlang manage concurrent execution.

Q & A


I DON'T ALWAYS HAVE A VALID ARGUMENT
BUT WHEN I DO, IT'S LATER THAT DAY, IN THE SHOWER


42
DON'T FORGET
YOUR TOWEL
RiskGambits