

## Re: ANN: Kamaelia 0.2.0 released!

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2005-08/msg00922.html>

---

- *From:* Michael Sparks <[ms@xxxxxxxxxxxxx](mailto:ms@xxxxxxxxxxxxx)>
  - *Date:* Wed, 03 Aug 2005 22:16:37 +0200
- 

phil hunt wrote:

> On Wed, 03 Aug 2005 16:57:34 +0100, Michael Sparks <[michaels@xxxxxxxxxxxxx](mailto:michaels@xxxxxxxxxxxxx)>  
wrote:

>>> Is the audience programmers or  
>>> less technical people? A project that allows non-technical people  
>>> to build complex network applications is an ambitious one, but not

....

>>It's a little ambitious at this stage, yes.

> But it couldbe there eventually?

Could? Yes. Will? Can't say. I can agree it would be nice, and given time/resources (or someone sufficiently interested externally) then it may happen. (IMO there's no real reason that it couldn't happen aside from time/effort/resources)

>>> What sort of servers and clients?

>>Whatever you feel like. If you want a server to split and serve audio,  
>>you could do that.

> This is streaming audio, right? For non-streaming I can just use an  
> ftp or http server.

There's more to network servers and clients than just audio & video, or unidirectional download.

For example the visualisation/introspection tool is an example of a client server system. The server is the visualisation tool. It listens on a specified port waiting for a connection. The client connects and sends data to it about the internal structure, and the server displays this.

>>>> \* Quickly build Multicast based network servers and clients  
>>> Serving what? Could I use it, for example, to build an n-player  
>>> encrypted VoIP server to allow people to do conference calls over  
>>> the Internet?

>>

>>You could do that probably. (Though we don't have a component  
>>for audio capture (though a read file adaptor reading from /dev/audio  
>>might work depending on your platform I suppose) and audio  
>>encoding at the moment, so those would probably be the core

Re: ANN: Kamaelia 0.2.0 released!

>>components to integrate.

>

> That's a slightly worrying answer for me, worrying because it seems  
> I've misunderstood the nature of the project. I assumed that  
> components for audio capture, and related activities, would be at  
> the heart of the project.

\*Our\* main interest, at the moment, is in /delivery/ of content.

Dealing with capture would be largely re-inventing wheels before we know whether the framework is a suitable framework. We are looking at making it possible to use pymedia for dealing with capture/encoding/decoding.

There's a number of things along the way we need to deal with this, but we're not starting from the perspective of capture.

(After all for capture we can generally look at using dedicated encoder hardware that will often spit out it's encoded information in the form of a network connection. As a result capture and encoding hasn't been a priority as yet. Sometimes looking into a project from the outside I can appreciate that certain decisions might look strange, but consider that you don't need to worry about capture in order

>>> (I mean proper encryption here, the sort GCHQ or the NSA can't break)

>>

>>I'd be impressed if that could be written, using anything really. (Can't  
>>implies never)

>

> What -- good encryption? That's pretty much a well-known technique  
> these days (unless the NSA has some \*very\* advanced hardware in  
> their basement, which I strongly suspect they don't).

You said \*can't\*. That says to me cannot ever be broken. If you have a large number of listeners, as your statement implied, that implies decryptable by many listeners – you then just need one compromised listener (essentially you're asking for the equivalent of implementing a DRM system that the NSA couldn't break...).

If you can provide me with a library that you can guarantee that it will satisfy the following properties:

encoded = F(data)

and a piece of client code that can do this:

decoded = G(encoded)

Then yes, that can be wrapped. That's trivial in fact:

---(start)---

```
from magic import unbreakable_encryption
```

```
class encoder(component):
```

Re: ANN: Kamaelia 0.2.0 released!

Re: ANN: Kamaelia 0.2.0 released!

```
def __init__(self, **args):
self.encoder = unbreakable_encryption.encoder(**args)
def main(self):
while 1:
if self.dataReady("inbox"):
data = self.recv("inbox")
encoded = self.encoder.encode(data)
self.send(encoded, "outbox")
yield 1

class decoder(component):
def __init__(self, **args):
self.decoder = unbreakable_encryption.decoder(**args)
def main(self):
while 1:
if self.dataReady("inbox"):
data = self.recv("inbox")
decoded = self.decoder.decode(data)
self.send(decoded, "outbox")
yield 1
---(end)---
```

If you believe you can implement F&G for general use such that F&G can //never// be decrypted by anyone other than authorised recipients, I suggest you cease this conversation – you have some highly marketable technology.

```
>>>>The basic underlying metaphor of a component us like an office worker
>>>>with inboxes and outboxes, with deliveries occuring between desks,
>>
>>> That metaphor brings up an image (at least to me) that the sorts of
>>> data that can be communicated are things like documents,
>>> spreadsheets, business graphs, memos.
>>
>>>They could indeed. The underlying framework doesn't differentiate
>>>between data nor have any realtime aspect embedded in the system
>>>at present. Just because we're focussing on systems that have a realtime
>>>element and are multimedia based, this does not mean the system is
>>>limited to that.
>
> Again, this makes me think I've misunderstood the project.
```

Realtime systems are a subset of all systems that are interesting in terms of network delivery & multimedia systems. Realtime scheduling is a well known area and if/when this becomes an issue, we'll look at adding it into the mix. The real problem is dealing with concurrency and making it simple to work with. Making realtime concurrent systems easy to work with strikes me as running before you can walk.

(Network systems are naturally concurrent, so if you're aiming to make network systems easy to build you're really talking about making concurrent

Re: ANN: Kamaelia 0.2.0 released!

systems easy to build.)

>>> OK, I get the straming part of it. But what asbout non-streaming  
>>> stuff? What other protocols are necessary?

>>

>>One example is peer to peer mesh setup. People normally  
>>think of P2P as a distribution mechanism. However, the underlying  
>>approach also very good at setting up communications meshes.

>

> When you say a mesh, what do you mean?

I mean a mesh. (see below)

>>This could be of use in many areas, such as GRID based systems  
>>for distributed rendering, application layer multicast, and network  
>>multicast island joining.  
> Unpack, please.

They're all big systems, all of which utilise networks of collaborating systems for different purposes.

Grid starting point:

\* [http://en.wikipedia.org/wiki/Grid\\_computing](http://en.wikipedia.org/wiki/Grid_computing)

\*short\* introduction to application level multicast (has many names, including overlay multicast):

\* <http://www.mnlab.cs.depaul.edu/seminar/fall2002/Overcast.pdf>

\* Also puts the term "mesh" in context.

Multicast island joining is a special case and is exactly what it says – joining multicast islands together.

>>Due to the illegal /uses/ of P2P, much work in this area is difficult to  
>>reuse due to defensive coding.

>

> Oh. Could you give an example?

How many RFCs have you seen documenting the protocols used by (say) Napster, Bit Torrent, Gnutella, Limewire, Freenet? The legitimate uses of Bit Torrent for example tend to get ignored by certain large companies when trying to shut down systems.

>>We also have to be able to demonstrate system to other people  
>>inside the BBC in a way non-technical people understand. That means  
>>showing structures in a friendly dynamic way, showing pictures,  
>>playing sounds (hence visualisation – looking inside running systems).

>

> Visualisation, if done properly, ought to be useful to technical  
> people too.

It is (as mention on the page describing the visualisation tool).

Regards,

Michael.

---

• **References:**

- ◆ **ANN: Kamaelia 0.2.0 released!**  
◇ *From:* Michael Sparks
  - ◆ **Re: ANN: Kamaelia 0.2.0 released!**  
◇ *From:* phil hunt
  - ◆ **Re: ANN: Kamaelia 0.2.0 released!**  
◇ *From:* Michael Sparks
  - ◆ **Re: ANN: Kamaelia 0.2.0 released!**  
◇ *From:* phil hunt
  - ◆ **Re: ANN: Kamaelia 0.2.0 released!**  
◇ *From:* Michael Sparks
  - ◆ **Re: ANN: Kamaelia 0.2.0 released!**  
◇ *From:* phil hunt
- Prev by Date: **Re: pain**
  - Next by Date: **Re: Wheel–reinvention with Python**
  - Previous by thread: **Re: ANN: Kamaelia 0.2.0 released!**
  - Next by thread: **Re: ANN: Kamaelia 0.2.0 released!**
  - Index(es):
    - ◆ **Date**
    - ◆ **Thread**