# Behind the Scenes of Justin.tv

Posted At : August 3, 2007 4:35 PM | Posted By : Stefan Richter
Related Categories: FMS, Site Check

You know Justin, don't you? The guy who's broadcasting himself 24/7 with a camera strapped to his head. His site justin.tv has recently started a transformation into a live broadcasting platform for people like you and me but is for now limited to a selected set of broadcasters with the intention of opening it up to a broader audience - reminds me a bit of ustream.tv.

Having blogged about justin.tv once before I received an email from Kyle Vogt, the main technical brain behind the operations at justin.tv. Kyle was kind enough to lend me a few minutes of his time for a short interview which I'll try to summarize now.

What's obvious right away when looking at justin.tv is the fact that the video feeds are delivered in Flash. That's significant when you realize that the site has been around for a fair few months, going back to early 2007 when very few people were attempting to stream live Flash video on a large scale. There were very few CDNs that supported live Flash video - in fact at the time when justin.tv went on air there was only VitalStream - so the choices for justin and his team were limited. And VitalStream they chose - with mixed results. In a nutshell, it didn't scale. Justin.tv has a very spiky traffic pattern and when the spikes went up rather than down the team realized that VitalStream didn't handle th eload very well and users kept being cut off time and time again.

The next attempt involved Wowza, a lower cost alternative to Flash Media Server, and a custom built, Python based stream replicator which Kyle developed. This stream replicator used to take the incoming live feed and pushed it out to 5 Wowza boxes. The setup worked ok to an extend, but another wall was being hit when some features could not be implemented due to Wowza being a closed source platform. The team had to try their third approach.

Again Python was utilized to build a custom server alongside a custom protocol which the server instances use to communicate with one another. These server clusters contain multiple origins and are able to redirect streams depending on load, something that is according to Kyle not possible with existing servers and licensing.
The protocol to send and receive the Flash video stream is of course RTMP - but this protocol is converted by Python into their own custom built protocol and then back again to RTMP before it hits the Flash Player. Neat! Moreover, all streams are also recorded and archived.
Deployment is handled over Amazon EC2, the Amazon Elastic Compute Cloud, a system which allows for immense scalability at the push of a button.

The text chat is also not what it first appears to be. In fact it's simply a Flash UI for an IRC based chat system. these guys clearly know how to build a solid and hugely scalable system on a budget.
The streaming operations run 24/7 nonstop, and result in a monthly data transfer of around 10 terabyte. Peak throughout is somewhere around 1 GBit/sec.

So there you go, large scale Flash video broadcasts on a less than ordinary backend. I

think what these guys have pulled off is genius.