Botball Live: Bringing Real-Time Web-Based News Coverage and Archiving to Botball Events
Jeremy Rand, Stuart Brothers, Abdullah Sahyouni, Lena Nadolinski, Wesley Myers
Norman Advanced Robotics, Explorer Post 1010, W.T. Woodson High School, W.T. Woodson High School, Carnegie Mellon University
jeremy.rand@ou.edu, lamarer88@yahoo.com, abdullahsahyouni@gmail.com, ideafuser2011@gmail.com, wmyers@andrew.cmu.edu

Botball Live: Bringing Real-Time Web-Based News Coverage and Archiving to Botball Events

1 Introduction

As robotics education becomes more popular, Botball is quickly expanding. New regions are popping up, and existing regions are becoming more competitive. At the same time, the economy remains tight, and many team members and supporters are unable to travel to tournaments with their teams. We feel that this combination has unfortunate effects, since not being at the tournament inevitably decreases the excitement, and therefore decreases the motivation to participate in or support Botball the following year. What is needed is a way for team members and supporters to keep up with the tournaments, without traveling. What is needed is Botball Live.

Botball Live is a platform we developed for the Botball Youth Advisory Council, which aims to provide real-time web-based news coverage of Botball events, maybe even with built-in archiving. It was first developed for the Oklahoma regional (but not used due to a last-minute hardware failure), and then successfully deployed in a new form for the Greater DC regional. We expect to deploy a further enhanced version at GCER 2011. This paper will describe the history and current status of the project, and where we think it's going.

2 Attempt 1: Oklahoma Regional

In December 2010, we began work on Botball Live with the intent of utilizing it at the Oklahoma regional. The work was divided between the video broadcast/archiving system, the score Tweeter, and the Web dashboard.

2.1 Video Broadcast/Archiving Via VLC

We planned to have one video capture device per table connected to a central PC. The PC would be running the VLC software [1]. VLC contains a feature called VLM, which, among other things, allows multiple streams to be handled simultaneously. VLM can also be controlled via an HTTP interface. We configured VLM to output its streams via the network, from which we could broadcast. At the same time, the VLM HTTP server listened for commands via the private tournament WiFi network.

The CBC's at the game tables also were connected to WiFi, and were programmed to ping a URL using GNU Wget (a command-line tool for fetching web pages) [8] whenever games

started/stopped. This URL pointed to the VLC HTTP server, thus making VLC aware of the timing of games. The URL contained an embedded command that would enable/disable logging of games to files, and automatically organized the filenames based on game number (entered by a judge into the CBC at the start of the game).

We didn't want viewers directly connecting to our VLC stream since we didn't have much bandwidth, and the Oklahoma tournament venue was unlikely to grant us access to incoming connections behind their firewall. After much searching, we discovered that Justin.tv [2] allows VLC streams to be broadcast through their servers. We thought this would be awesome, until we tried the software and found that it didn't work on our computers. There was also no documentation, and Justin.tv did not return our support requests. Eventually we stumbled across a blog post [3] with instructions for publishing ffmpeg streams to Ustream [4]. Based on the post, we could tell that ffmpeg was being instructed to emulate the Flash Media Live Encoder, and that therefore this would work with minor modification to connect to Justin.tv as well, giving us a choice between Ustream and Justin.tv which we planned to work out later. After much reading of the ffmpeg manual, we found how to get ffmpeg to read VLC's stream, encode to h.264, and broadcast to Justin.tv. For this rather complex setup, we found Justin.tv somewhat easier than Ustream, because Justin.tv begins broadcasting immediately, while Ustream expects a button to be manually clicked on their website when going through this process.

This setup allowed video to be archived by match, and broadcast via Justin.tv to anyone who wanted to watch it. We moved on to the next portion of the project.

2.2 Score Tweeter

There's limited use in seeing a video of the match if one doesn't know what the score was. We developed some Bash scripts that were capable of posting scores to the Botball Twitter feed, with the intent of hooking them into Joust, the KIPR bracket software. They were also designed to be manually usable, so if a volunteer was available to enter scores, they could be posted without Joust integration, thus hedging our bets on actually getting the system to work.

2.3 Web Dashboard

We built a Botball Live page on the Botball Community website [5] which embedded the streams, the Twitter feed, and the Botballer's Chat (which we intended to use both for encouraging community discussion and for allowing technical problems to be reported). One challenge was that most users probably wouldn't want to watch all six streams, the Twitter feed, and the chat room simultaneously, and limited screen real estate was available (particularly due to the fixed-width Botball Community layout). To solve this, we implemented a "Widget" system which allowed users to only display the items they wanted to see. The Widgets were also designed to allow users to watch multiple tournaments simultaneously (a few of the regional tournaments overlapped). The end result wasn't the prettiest website out there, but it worked. With that, we ran several successful tests of the system, and were ready for Oklahoma regionals. Or so we thought....

2.4 Failure in Oklahoma

We were informed by KIPR that they were only able to stream one table due to lack of hardware, and that we would have to provide the video capture device (luckily, we had one lying around).

They also informed us that they weren't able to find a volunteer to enter Twitter updates (nor had they been able to provide us with the Joust source code), and since the only Oklahoma YAC member was competing on a team, we had to scrap the Score Tweeter.

Nevertheless, we were hopeful that we would be able to do a pilot run, and expand later. Things went well as we showed KIPR how to operate the software; we were told that KIPR would be installing our software on a desktop they had lying around, and that if they had issues making it work, we could work it out with them during the lunch break on tournament day. We went home quite satisfied.

Then, we got these e-mails from KIPR:

> 11:11PM: "I've run in to some problems getting Windows going on the computer I was planning to bring. It looks like my drive is failing. I'm going to see what else I can scrounge together, I'll let you know."
>
> 12:44AM: "Ok, it looks like I'm not going to be able to get this working tonight. I'm not sure what's wrong and I'm pretty much out of time."

Without a working computer, the project was toast for the Oklahoma regional. We decided that we would try again at a subsequent regional.

3 Attempt 2: Greater DC Regional

We had three YAC members in the Greater DC region, so we decided to make it our next target. However, we decided to try a different architecture due to lack of hardware. The three YAC members didn't have any video capture devices, but did have access to large numbers of iPod Touches and iPhones, so we decided to simply use the iOS Ustream app. This was a much simpler setup than VLC, and although the feature set was smaller, it had a major cost advantage -- a computer with the ability to encode six h.264 streams would have been expensive to obtain for our YAC member there, whereas the iOS devices (with their hardware video encoders) could easily stream the tables with quality that was actually higher than the VLC setup. We were okay with the loss of match archiving for the moment; at this point, we just wanted something to work at all.

The DC tournament venue was somewhat more restrictive than the Oklahoma venue, and did not allow Ustream traffic through its firewall. (In fairness, we contacted Norman High School's IT department several weeks in advance, whereas by the time we were ready to go for DC, we only had 48 hours before the tournament and were not able to reach the Woodson High School IT department.) Luckily, we had a 4G hotspot lying around, and were able to make it work with the iOS devices. Unfortunately, the 4G bandwidth limited us to two streams.

We successfully broadcast the DC regional, and used Ustream's text-based update feature to similar effect as the Score Tweeter was intended to provide (we manually entered the information, so it was less thorough and immediate than it could have been). KIPR and YAC officials announced early in the tournament that the tables were streaming, and gave the URL (with the request for the audience to please let all their friends know). We also announced the

event on the front page of the Botball Community 48 hours in advance.

We did encounter a few technical issues when the iOS devices tried to switch networks of their own accord, and were delighted to find that we immediately received notification in the Botballer's Chat -- people were actually watching! The issues were fixed, usually within 5 minutes, but we did have one case where it took us about 45 minutes to get a stream back online. During the tournament, our maximum number of simultaneous viewers was 25 (we don't have statistics for the cumulative number of unique viewers).

After the tournament, we talked to KIPR officials, who were highly impressed with our success, and wanted to help us run Botball Live at GCER 2011. Naturally, our response was a big "Absolutely!"

4 GCER 2011 Botball Live Plans

We intend to use a system at GCER somewhat similar to what we had at the Greater DC regional, but with some improvements. A brief (non-thorough) list of improvements we're considering is:

- Re-adopt the Score Tweeter, and utilize a similar system with Joust integration to show team names in a caption under the streams.
- Record the matches on the iOS device side. Something like the Cydia app Display Recorder [6] would probably be suitable for this. Even if the matches cannot be cut in real-time, the game lights CBC's could be rigged to generate timestamps which would allow automatic cutting after-the-fact.
- Stream all events, not just two tables of D.E. Specifically, the following events would be streamed:
    - Botball Matches
        - Seeding
        - D.E. Qualifying Founds
        - D.E. Final Rounds
        - Alliance
    - KIPR Open Matches
        - Seeding
        - D.E.
    - Conference Events
        - Featured Speakers
        - Teacher Track
        - Student Breakout Sessions
        - Feedback Session
        - Awards Ceremony
    - Open Practice Tables (Lower Priority)
    - (The Autonomous Robot Showcase and the Student Activities would not be streamed.)
- Broadcast bonus footage, e.g. live interviews with team members, presenters, and YAC and KIPR representatives. This would require additional manpower.
- Link the iOS devices with the PA system, so that the audio quality is higher for

the broadcast (at DC, the audio was extremely echoed due to the built-in iTouch microphones).

- Connect the iOS devices to our PC via VNC, so that we can fix technical problems faster. 45 minutes is unacceptable. Similarly, have YAC/KIPR technicians keep an eye on the Botballer's Chat in shifts, so that we're immediately aware of technical problems without making someone sit there for hours (which is what we did at the DC regional).
- Redo the Web Dashboard to have more AJAX / Web 2.0 functionality. This would allow the GCER schedule to update in real-time, and allow the Widgets to be dragged/zoomed arbitrarily.
- Upload Joust files to a Botball Live Widget as they are updated, further allowing users to see what is going on.

A potential issue is network access and bandwidth. We plan to use three 4G hotspots at GCER, but if the conference center building absorbs cell signals or bans 4G usage, we may have to improvise. Purchasing Internet access from the conference center is a possibility, albeit somewhat expensive. The conference center charges $10/day/device, so the best-case scenario for six streams through the conference center is $60/day * 6 days = $360 for Internet. A 4G hotspot is only $45/month, making the cost of Internet $135 for the conference. (A 4G hotspot also costs $99 per hotspot, but this is a one-time cost which serves as an investment for future seasons.) Each 4G hotspot can handle 10GB of bandwidth per month, which translates to 44 hours per iOS device; we will only need 36 hours per iOS device for the conference.

We think we're relatively likely to get a lot of these improvements implemented for GCER 2011. Will we get them all? Will there be unexpected surprised that require a re-working of the system? Only time will tell.

5 GCER 2011 Live-Blog Challenge

Last year YAC launched the GCER Live-Blog Challenge, which encourages teams to live-blog the conference and tournament activities. This year, we wanted to strongly integrate the Live-Blog Challenge with Botball Live. The plan was to aggregate all of the blogs' RSS feeds into the Community's content database, with user permissions managed by the teams operating the blogs. If you were logged into the Community, and a team had given you permission to see their live-blog, a special Botball Live widget would be available, which would display the live-blog contents, refreshing in real-time.

Unfortunately, due to technical issues on KIPR's side, we were unable to do the integration with the Botball Live Dashboard which we wanted. Nevertheless, we will operate the Live-Blog Challenge this year. The setup will be near-identical to last year, which was quite successful for a first-time attempt.

There were two issues that came up with the 2010 Live-Blog Challenge, which we would like to address here.

First, the projected date for announcing winners was not met, mainly because everyone was on vacation and it was impossible to get the whole YAC into a Skype chat at once as we had hoped. In the end, two YAC members were responsible for the final rankings since we wanted to get

the results out there without overshooting our projection any more than we already had (we projected 3 weeks, it took 2 months). We also had to deal with a particularly impatient team (we won't name them here) which frequently complained about the delay (calling it "unacceptable"). Perhaps that team is unaware that YAC is entirely volunteer-run, and that we didn't owe them prompt results. If they'd like to hire a dedicated team to manage and judge the Live-Blog Challenge, we'd love to talk to them; otherwise, they need to simply wait for us to get to it. Since we didn't like getting those complaints e-mailed to us, we are making a change: the projected date of announcing results is "Before the first regional of the 2012 season." Since we are confident that we will be able to get things done before that point (or at least we hope so), we anticipate that this will eliminate such complaints and make our work far more pleasant.

Second, a team (again, we won't name them) accused YAC of rigging the results in favor of Norman Advanced Robotics, which took first place, on the grounds that since Norman was represented on YAC, any victory by Norman could not be legitimate. Perhaps that team did not read our GCER paper last year on the issue [7], which clearly stated that "Since some of the judges are YAC members whose teams may be competing in the Live-Blog Challenge, judges must not include their own school in their ranking." This policy was adhered to; the judge responsible for Norman getting first place was not from Norman. We will not be changing any policies here for several reasons. First, many excellent teams are represented on YAC, and banning them from entering the Live-Blog Challenge would lower the average quality of the entries. Second, YAC is understaffed and we do not want to incentivize not joining YAC due to inability to compete in certain GCER activities. Third, we do not see any benefit to rewarding the poor sportsmanship involved in accusing a team of cheating with no evidence whatsoever.

Despite those minor issues, the 2010 Live-Blog Challenge went very well, and we hope to improve our 2010 enrollment of five teams for the 2011 Live-Blog Challenge. We look forward to seeing the blogs which teams come up with.

6 Conclusion

We think Botball Live has the potential to greatly increase the interest in Botball and GCER. Naturally, we will continue to work on the project next year. Do you have suggestions for Botball Live, or other activities YAC should undertake to improve Botball? YAC can be reached at the Botball Community [5]. From all of us at YAC, we wish you an awesome GCER!

References

[1] VideoLAN. VLC Player. http://www.videolan.org/vlc/ , April 2011.
[2] Justin.tv Inc. Justin.tv. http://www.justin.tv/ , June 2011.
[3] kyriii. Free Live Video Streaming with HTTP Live Streaming, uStream and justin.tv in a GNU Linux Environment. http://blog.kyri0s.org/post/1406637341/free-live-video-streaming-with-http-live-streaming , October 2010.
[4] Ustream Inc. Ustream. http://www.ustream.tv/ , June 2011.
[5] Botball Youth Advisory Council. Botball Community. http://community.botball.org , June 2011.
[6] R. Petrich. Cydia App: Display Recorder. http://www.ijailbreak.com/applications/cydia-app-display-recorder/ , May 2010.

[7] B. McDorman, J. Rand, M. Thompson. GCERSync and the Live-Blog Challenge: Preserving and Archiving the GCER Experience. Proceedings of the 2010 Global Conference on Educational Robotics, July 2010.
[8] Free Software Foundation. GNU Wget. http://www.gnu.org/software/wget/ , November 2010.