GENI Cinema: An OpenFlow-Based Live Video Streaming Service

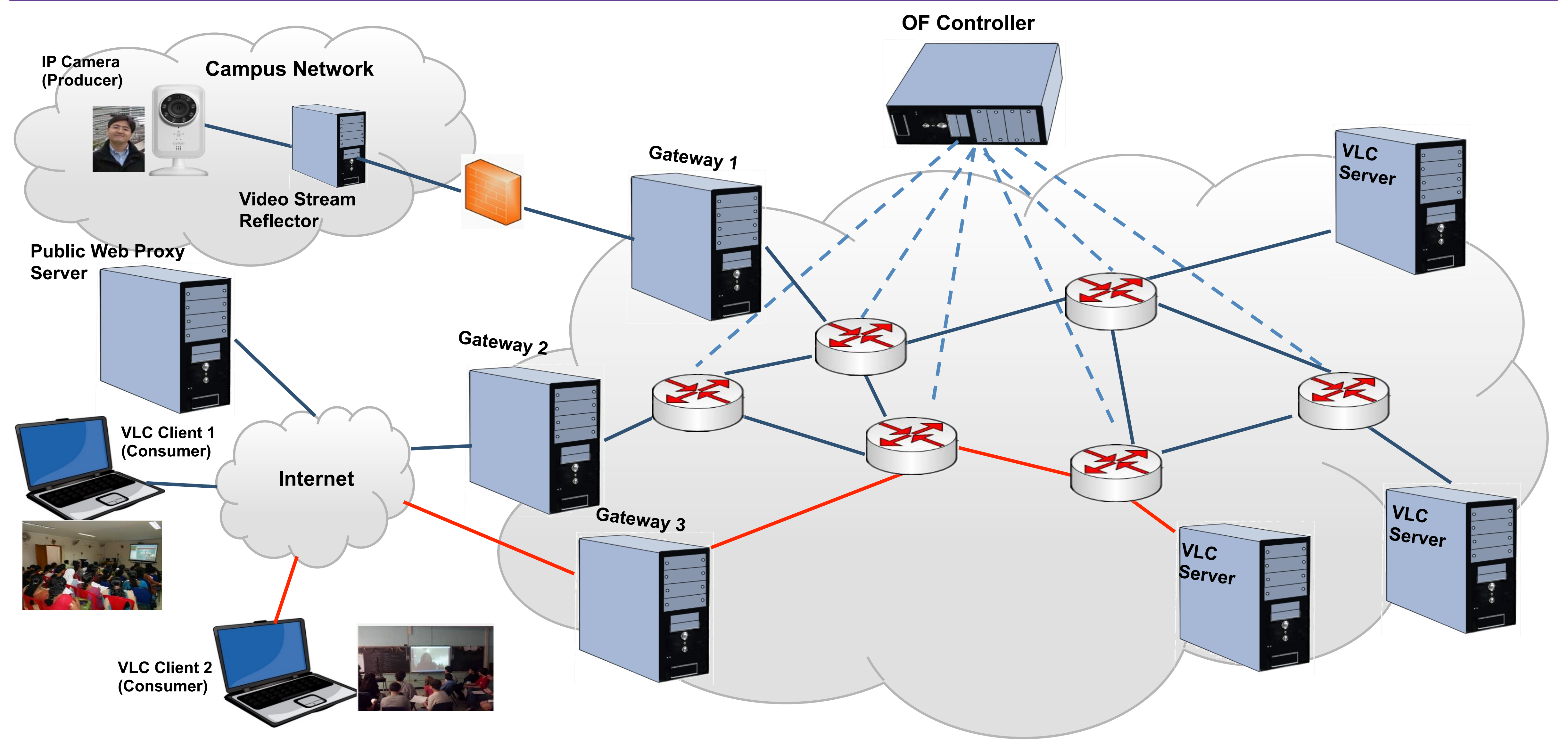
Ryan Izard, Qing Wang, Benton Kribbs, Joe Porter, Kuang-Ching Wang
{rizard, qw, bkribbs, jvporte, kwang}@clemson.edu
Aditya Prakash, Parmesh Ramanathan
aprakash6@wisc.edu, parmesh@ece.wisc.edu

Goals and Motivation

Video streaming over the Internet, be it static or live streaming, is rapidly increasing in popularity. Many video streaming services exist to serve a variety of needs, such as video conferencing, entertainment, education, and the broadcast of live events. These services rely heavily on the server application to adapt to increasing and decreasing demand for a particular video resource. Furthermore, they require the reallocation of resources and the restart of the stream when a client stops, starts, and/or switches to a different stream.

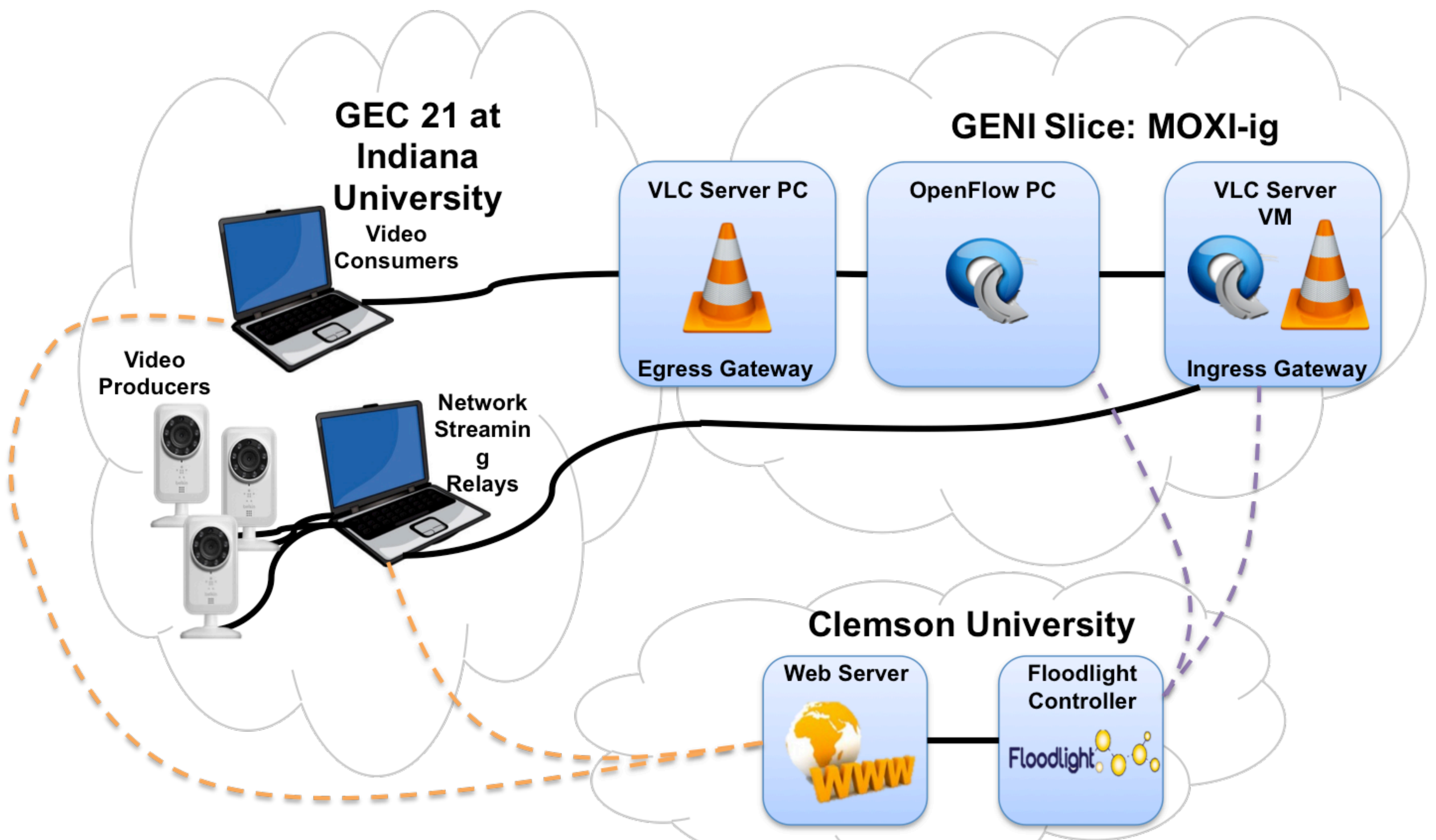
Our goal is to provide a scalable service for GENI that supports the broadcast of live video streams from an arbitrary number of video-producers to an arbitrary number of video-consumers, where video-consumers can change "channels" without disrupting their existing stream and without affecting the load on a particular video stream source.

GENI Cinema Architecture



- GENI aggregates/racks used for compute and network resources
- Public web server allows users to access the service to provide, browse, and watch live videos
- Floodlight controller maintains the running state of GENI Cinema and serves as a global resource arbiter for ingress and egress video streams.
- Web server communicates with Floodlight controller via JSON over REST API
  - Query for video availability and service status
  - Select ingress gateway for video producers
  - Select egress gateway for video consumers and handle channel changes
- Floodlight controller uses OpenFlow 1.3 group and bucket features to (re)route video streams to clients on demand.

Demonstration Scenario



Video Upload Procedure:
1) RTSP video streaming from camera
2) Stream relayed to gateway
3) Gateway relays video stream to back-end VLC video server
4) Floodlight disables video stream until clients connect

Video Request Procedure:
1) Client connects to public web server and queries for, and selects video of interest.
2) Floodlight checks and modifies OpenFlow switches to send selected stream to client
3) Floodlight sends address and port to website where the client's video stream will be available
4) Website connects and displays HLS stream for client, or website provides client with gateway IP and port to manually view the video with its own application (e.g. VLC)

- Network streaming relays at Indiana
- InstaGENI rack with Gateway, OpenFlow, and VLC machines
- Web server hosted at Clemson for video selection and streaming
  - Buffered HLS in browser
  - Non-buffered stream direct to VLC client
- OpenFlow 1.3 version of Floodlight controller running locally

Future Work

As network people, we are not well-versed in web design. We are presently investigating how to stream a non-HLS stream to a web browser using a free video player or natively with HTML5. HLS has proven to work well; however, due to the buffering of video time segments, when a client changes channels, the change is delayed by the size of the buffer. This delay is approximately 1 minute, which is unacceptable. With a non-HLS video stream with no extensive buffering, a channel change occurs within a few seconds after initiating the change on the web server; however, we have only been able to get this to work using VLC to open the stream at the client. An ideal solution would work within the browser.

On the OpenFlow side, we will further address scalability by allowing video streams to be assigned and split to multiple Open vSwitch machines from the host VLC servers. In this way, the video streams can be allocated to clients and load-balanced across as many Open vSwitch machines as the aggregate can support. The controller will also be improved in tandem with the website to support automatic resource reallocation in the event a video-consumer disconnects without notification. Finally, tests will be performed across multiple GENI aggregates. The controller and website presently support this, but this feature has not been tested extensively.