Optimizing High Latency Links in the Developing World

Yaw Anokwa, Colin Dixon, Gaetano Borriello, Tapan Parikh
University of Washington, University of California at Berkeley

It's always tough be the last speaker in a workshop, but I'll try to make it as painless as possible...

My name is Yaw Anokwa, I'm a third year computer science PhD student at UW. This "Optimizing High Latency Links" work was done with Colin Dixon (a UW grad student in networking) and my advisors, Gaetano Borriello at UW and Tapan Parikh at Berkeley.

This is an invited paper, so it's all work in progress. If you have any feedback, I'd love to hear it. Also, feel free to ask questions during the talk -- I'd love to hear people challenge my assumptions.

I'll just go through some of our motivation, challenges, the previous work and what we are working now.

This picture is taken in a small town called Rwinkwavu in Rwanda, a tiny landlocked country in East Africa.

Rwinkwavu is in the middle of nowhere (intermittent power, water, cell, etc), but it is a beautiful place and if you are looking for a quiet place with intermittent everything, it’s the place to be!

I spent six months in Rwinkwavu last year at a hospital network run by Partners in Health. The two hospitals and five clinics host a model for a comprehensive healthcare program that is being scaled up nationwide.

This is exciting work that is done with the Ministry of Health and the Clinton and Gates Foundations.

As part of the scale up, some Internet connectivity has been deployed. Because Rwanda is land locked most of these connections are satellite based which then connect to a wifi network in the hospitals.

The connectivity is used to communicate over the web, and to implement an electronic medical record system (called OpenMRS) for tracking all the HIV/TB patients.

The guy who manages the connectivity at all the sites is called $F, and he's definitely one of the best network admins in the country. He is responsible for all computing and networking infrastructure and he's had quite a bit of training in South Africa. He is basically the local champion.

That said, when I arrived in Rwinkwavu, the connection was managed by an old Belkin router as the fancy Cisco crashed regularly and was just too hard to configure. There was no caching or QoS implemented on the network.

Part of my job was to make this situation better and do it in a sustainable way, so it would work after I or $F left.

This paper is really driven by my experiences in Rwinkwavu and so let's start with what big challenges were.

Challenges

- Latency is a problem for networks whose usage patterns tend to swamp the low bandwidth, high latency links.
- Providing users with adequate service requires training and tools that are not easily found in low income regions.

Almost all the traffic we see in Rwink is web traffic and the occasional Skype call. VSAT is pretty latent as is and because of the lack of QoS it was pretty easy for one long lived connection to kill all the short connections by sticking them into TCP slow start.

It's not so much the throughput that was the problem, but the latency and usage patterns meant the links get swamped. And as I am sure you know that just makes the connection unusable. We got a lot of complaints.

In addition to latency, providing the best service goes beyond just getting connectivity. The amount of training required use Squid or configure a Cisco router and do other network optimizations was just beyond what the capabilities of our staff had.

We don't think our deployment is unique. Low bandwidth, high latency connections are the standard in the developing world. And I would say most of the staff that run these networks aren't well trained.

Previous Work

- You can solve latency with TCP hacks, proxies, caches, time shifting, delay tolerant networking, ad blocking, compression, etc.
- Problem? Research code, non-interactive apps, no media, etc.

So there is a lot of work on the first challenge of latency. Neil Spring at UW at InfoCom 2000 addresses some of these issues in his ‘Receiver based management of low bandwidth access links’. Really good read if you get the chance.

There is also work on TCP optimizations, proxies of all sorts, delay tolerant networking, ad blocking, compression, etc. You can see the paper for some of the references. Some of this can be done in parallel with a solution, but we need something deployable and realtime.

Everyone uses Facebook and Gmail...

Previous Work

- You can "solve" sustainablility with apathy, frustration, trial and error, Yahoo! Answers, WNDW, home router.
- Problem? Apathy, frustration, trial and error, Yahoo! Answers.

The second challenge of a deployable system is a little harder.

Worst case is basically giving up or getting angry. Some of our admins uses Yahoo! Answers to try to figure out. At best you can get a home router, but that gives you an iptables setup that is generally used for asymmetric links and is pretty outdated.

In our case, there was a fair amount of trial and error as well as trying to use online texts. Bottom line? Too much work for such a hectic environment. We have power failures and networks to keep running.

We need something that goes beyond research code. Something that requires little documentation and a great interface. Something my parents could use. We need something that $F can deploy and maintain.

Latency

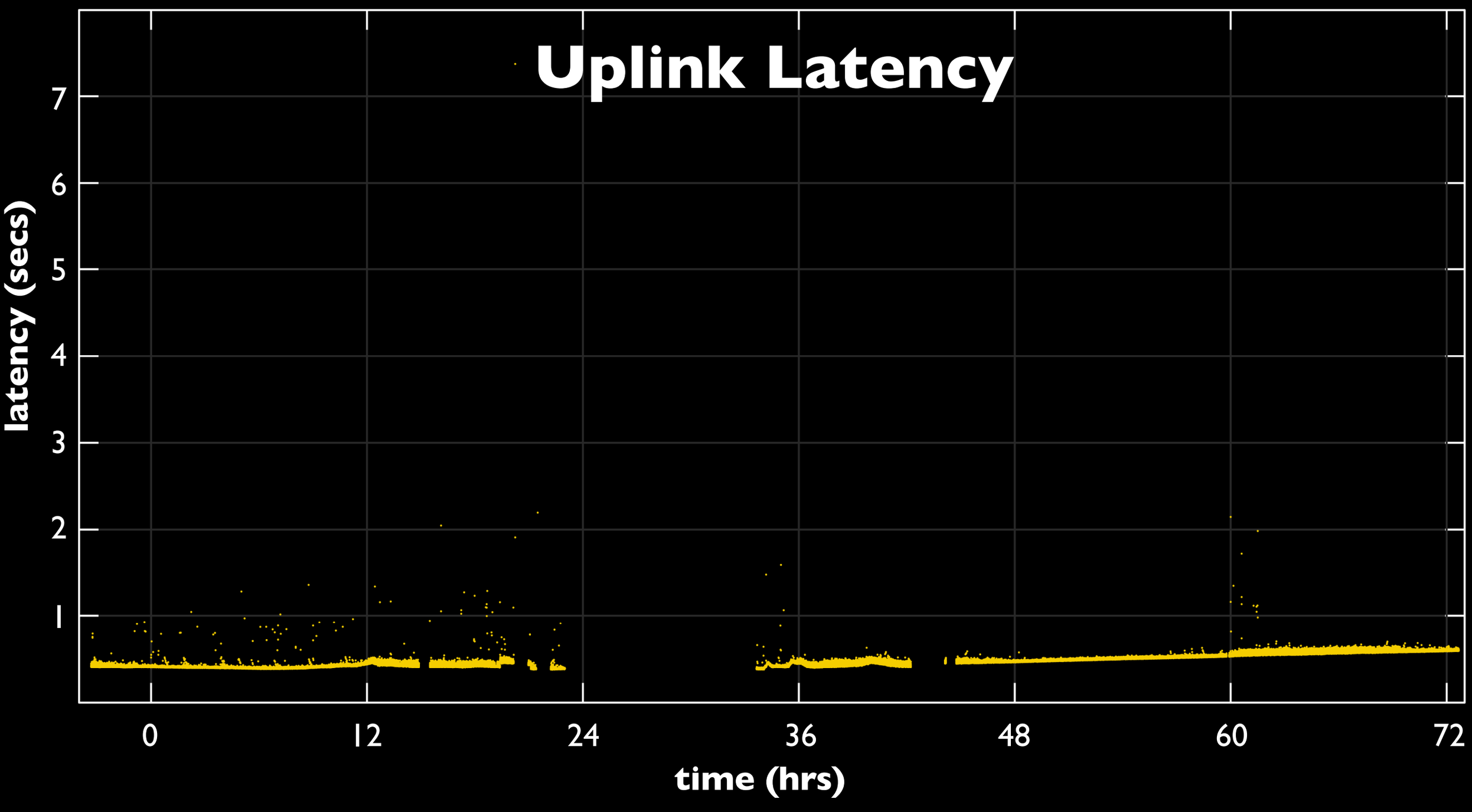
Measurements

- VSAT with 700 kbps down, 200 kbps up.
- UDP packets from Seattle to Rwanda over 72+ hours. The uplink and downlink were sampled separately.
- Some clock skew in the results, but the signal is much larger than this noise.

As a quick demonstration of the problem, we did a few quick measurements.

The VSAT does 700 kbps down and 200 kbps up. Traffic goes from Rwanda to Israel then on to the internet.

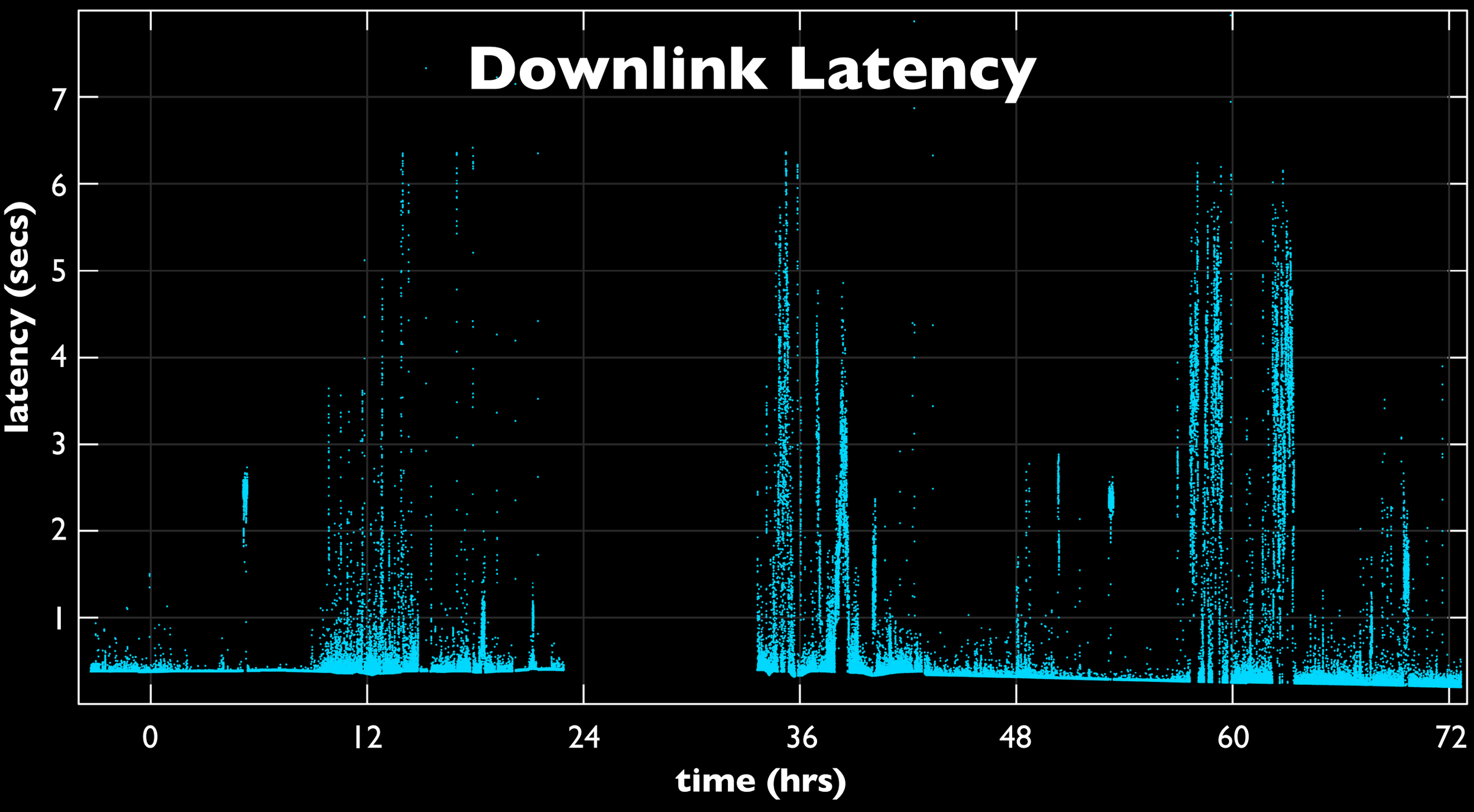
To test things, we sent UDP packets between Rwanda and Seattle. We sampled the uplink and downlink separately and although some clock skew, but it didn't change the facts of the problem



On the y-axis is the latency in seconds and the x-axis is the time.

We did this across a three day period. Hours 12,36,60 is about 10 am in the morning. This gap is from the connection going down -- which happens about once a week.

Uplink: Average delay is 478 ms, minimum is 376 ms. Nothing very exciting.



Average delay is 572 ms with the same 376 ms minimum. You can see these sustained 2-5 second latencies during working hours.

The spikes are mostly due to long lived downloads. Either attachments or some heavy objects on web sites. This kind of the meat of the problem and after thinking, this is a first cut at our solution.

QoS Approach

- Simple QoS prioritization be used to provide more behavioral classification of flows instead of protocol and port.
- Assume high bandwidth flows are more tolerant of high latency whereas low bandwidth flows are not.

Because all our traffic goes across port 80, classic port and protocol optimizations basically don't work.

We are doing more behavior-based classification. Our underlying assumption here is that high-bandwidth flows will be more tolerant of high-latency, whereas low-bandwidth flows will better benefit from low-latency.

This approach will likely work because of the workload we see in the developing world [Bowei Du, Mike Demmer @ WWW 06].

A lot of the object sizes we see are heavy-tailed, which implies that flow duration will be heavy tailed. Flows which have used a lot of bandwidth are likely to continue dominating the links. Such flows should be tolerant of extra delay.

QoS Approach

- QoS will divide flows into fast (flows which when grouped will not cause queueing) and slow (all others).
- All queuing is at one point before link. We ensure all flows are fed, but migrated to the appropriate class with sliding window estimate of throughput.

So we are going to look at two classes for flows: fast and slow.

The fast class will contain a number of the smallest flows such that they will not cause any noticeable queueing among themselves and so can achieve near the physical link latency.

The rest of the flows will be classified as slow and given no guarantees of reasonable latency.

An example of a fast flow is the individual HTTP requests for text on webpages, while an example of a slow flow is requests for large file downloads.

All the queuing is managed at one routing point and we are using a sliding window estimation of each flow's throughput to dynamically determine which class is each flow is in.

It's not fancy, but so far it has worked in our simulations.

Deployment

Devices

- Routing code is being written for Linux and the OpenWRT platforms. Targeting the Linksys WRT54G and Asus WL-500G.
- Put emphasis on device interface and documentation to ensure sustainability. Looking at tools like eBox, webmim, ClarkConnect.

Our target platform is Linux and OpenWRT which is open source firmware which runs on a bunch of routers.

We really like the Linksys WRT54g, which is a device that sparked this open router movement. Even better is the Asus WL-500G which has USB ports so you can attach a flash disk and do caching and routing.

Both of these are readily sourced, easy to upgrade and are just a lot more robust than a desktop PC in a dusty, dirty environment.

We are putting a emphasis on doing a lot of interface and documentation work to make sure the stuff is sustainable.

Status

- Core of routing code has been written and cross compiles to OpenWRT.
- Starting the push to do initial sketches on interface. Hope to take a bunch of sketches to Rwanda next time I'm there.
- Evaluate effectiveness on both the latency and usability of the system. Compare with high end similar routers.

Again, this is still a work in progress, we are working on two platforms. Colin is putting the finishing touch on the routing code.

We are starting to do the work on the interface. It will likely be based on the existing router interface. We have a plan to head to Rwanda and iterate through a bunch of user interfaces and documentation so we can deploy something that our staff can use.

Evaluation plan is to run real traffic through it and just see how comfortable $F is with the system. It may also be good to compare it to some more expensive stuff.

So with that, are there any questions?

Questions?

Really want to encourage everyone to really start getting in touch with real deployments and building real systems.