

UKAIRO: Internet-Scale Bandwidth Detouring

Thom Haddow, Sing Wang Ho, Cristian Lumezanu,
Moez Draief, Peter Pietzuch

February 10, 2011

Motivation

- ▶ Here we focus on **HTTP bandwidth**
 - ▶ HTTP is increasingly used as a generic transport protocol
 - ▶ Bandwidth is probably the most user-visible Internet performance metric
- ▶ Increasingly critical given the trend towards cloud-based storage architectures
- ▶ CDNs are excellent approach in certain cases, but...
 - ▶ Not all content is popular or statically cacheable.
 - ▶ CDNs generally cost *providers* to provision
- ▶ Can improve bandwidth through **detour routing**...

Detour routing

- ▶ **Detour routing:** Redirecting Internet traffic via tertiary hosts to improve upon the metrics of the default Internet path

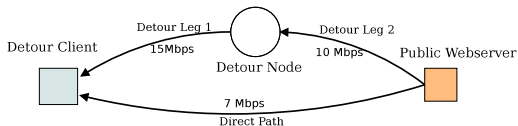


Figure: Bandwidth detour improving 7 Mbps direct path to 10 Mbps

- ▶ Does not necessarily require adaption of existing infrastructure or protocols
- ▶ Works for many metrics: latency, jitter, loss, availability...
- ▶ For bandwidth, large-scale measurements show 75% of paths could be improved by **at least 1Mbps and 20%**

Challenges

- ▶ **Discovering detour nodes**
 - ▶ Bandwidth is a very difficult metric to work with
 - ▶ Expensive to measure
 - ▶ Extremely unstable
 - ▶ *Scalably* identifying good detour nodes
- ▶ **Exploiting detour paths**
 - ▶ Choosing an appropriate detour node for a given client path
 - ▶ Transparently redirecting traffic
 - ▶ Obtaining high throughput via Internet edge nodes

Ukairo

- ▶ A service which discovers and exploits alternative detour paths to improve client performance to arbitrary Internet destinations

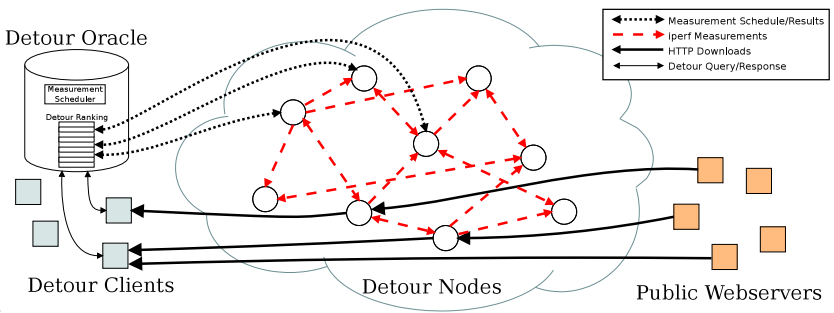


Figure: Ukairo architecture overview

Presentation Overview

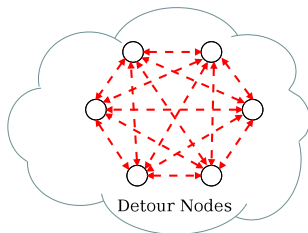
- ▶ Experimental platform
- ▶ Addressing challenges
 - ▶ Locating good detour nodes
 - ▶ Exploiting detour nodes
- ▶ Architecture overview
- ▶ Performance results using PlanetLab and public web servers
- ▶ Future work

Experimental Platform

- ▶ Current deployment platform is PlanetLab
 - ▶ Gives easy access to large set of nodes, but approach is generally applicable to any deployment platform.
- ▶ Used public webservers for some parts of evaluation
- ▶ Measurement analysis based on 217 node (*ie.* all sites) all-to-all TCP throughput measurements (iperf).
 - ▶ This is close as we can get to an “Internet scale” experiment
- ▶ Evaluation based on random selections from the above nodes

Locating detours

- ▶ **Given a large set of Internet nodes, how do we identify which can act as effective detours**
- ▶ All-to-all measurement works best, but not scalable

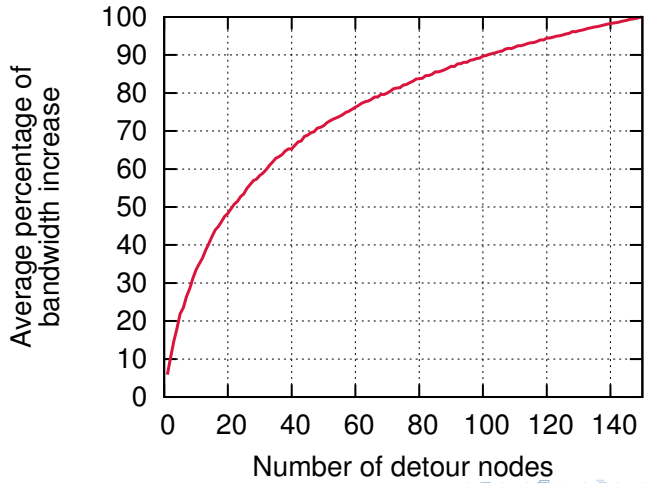


- ▶ Cannot use latency or lighter measurements as a predictor
- ▶ Have previously proposed system based on AS-path traces, but this does not work well for bandwidth.

Scalable detour discovery

- ▶ Observations from measurement analysis:
 - ▶ Given a set of nodes, a small minority of nodes contribute the vast majority of bandwidth improvements
 - ▶ Detour nodes are specialised in terms of which paths they can detour for, and there is much overlap
- ▶ Approach: **Limit number of detour nodes we use to perform path bandwidth measurements**
 - ▶ Select these nodes randomly
 - ▶ Can still identify a sufficient number of effective detour nodes
 - ▶ Reduces the unscalable overhead of all-to-all measurement to a fixed cost

Scalable detour discovery: Results



Scalable detour discovery: Conclusions

- ▶ A random selection of **50 nodes** can achieve around **75%** of the total potential detouring improvement
- ▶ Fixed maximum of $\sim 50^2$ measurements, but this can be reduced by intelligently choosing paths to measure [SWH]
- ▶ Outcome: **A ranked list of detour nodes which can be used to improve arbitrary Internet paths**
- ▶ Around half actually provide significant detouring improvements

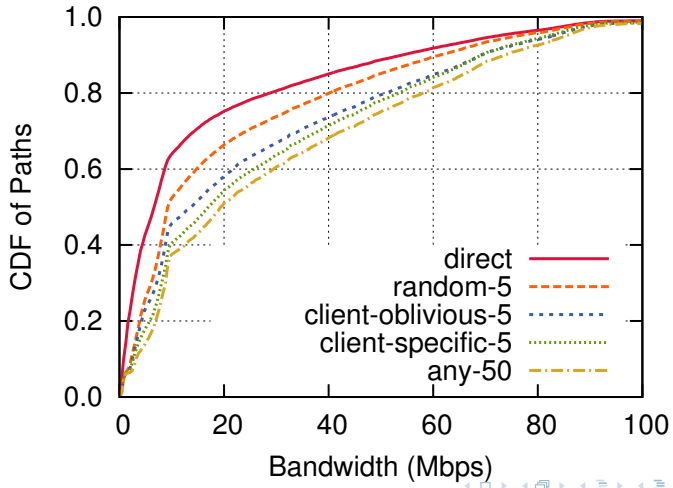
Selecting detours

- ▶ 25 detour nodes is too many to choose from
 - ▶ Need to reduce to the size of the detour set to a manageable size
- ▶ Observations from measurement analysis:
 - ▶ Any given client can achieve near optimal bandwidth improvement by employing **only 5 detour nodes**
 - ▶ These detour nodes are **specific to the client**
 - ▶ No obvious means of acquiring this mapping in a deployment context
- ▶ Evaluated various means of mapping 5 detour nodes to clients. . .

Mapping detour sets to clients

- ▶ client-specific-5
 - ▶ Choose the best 5 nodes for each client
 - ▶ Not feasible in a real deployment
- ▶ client-oblivious-5
 - ▶ Choose the same 5 best ranked node for each client
- ▶ load-balanced-5
 - ▶ Choose 5 nodes at random from the set of effective detours
- ▶ random-5
 - ▶ Choose 5 nodes at random from the set of 50

Mapping detour sets to clients: Results



Mapping detour sets to clients: Conclusions

- ▶ Random choice of detour set provides some improvement, but rank-based decisions significantly increase performance.
- ▶ The best possible client-specific mapping of detour nodes to clients provides performance close to that of using all available detour nodes
- ▶ Only employing the top 5 ranked detours provides excellent performance (but may spread load badly)
- ▶ We have also devised a method for dynamically adapting any clients' detour set towards its client-specific version [SWH]

Architecture components

- ▶ Detour oracle
 - ▶ Schedules measurements between detour nodes
 - ▶ Analyses measurements results and ranks detour nodes
 - ▶ Supplies lists of detours to detour clients
- ▶ Routing plane
 - ▶ SOCKS proxies are deployed on all detour nodes, enabling arbitrary TCP connections to be relayed via them
- ▶ Detour client
 - ▶ HTTP download library which uses adaptive detouring to accelerate large file transfers.

Architecture overview

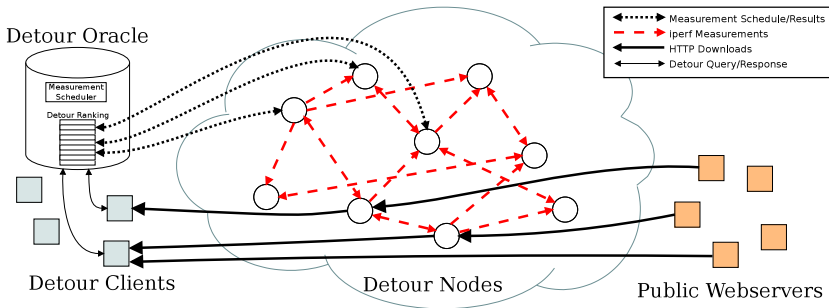


Figure: Ukairo architecture overview

Dynamic Path Selection

- ▶ Given a small set of potential detour nodes, how do we pick the right one for a given path?
- ▶ Should remain transparent to clients
- ▶ Bandwidth extremely dynamic — have to test detour effectiveness at point of use
- ▶ TODO

Dynamic Path Selection diagram

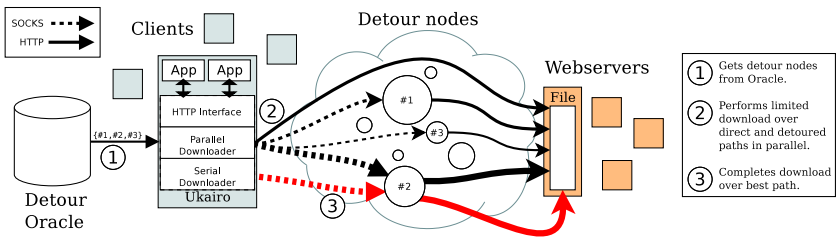
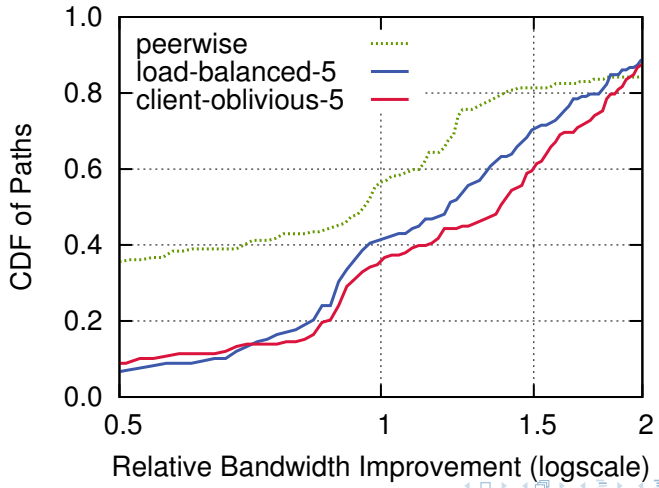


Figure: Ukairo architecture overview

Evaluation overview

- ▶ Prototype system with 50 detour nodes hosted on PlanetLab
- ▶ Set up clients to download 40MB test file from target systems
- ▶ First experiment evaluates all-to-all intra-PlanetLab performance with 10 clients and 25 servers all hosted on PlanetLab
 - ▶ Additionally, performance contrasted with the PeerWise latency detouring platform
- ▶ Second experiment evaluates performance with 10 PlanetLab clients and public webserver targets (100 Linux kernel mirror archives)
- ▶ Final experiment evaluates intra-PlanetLab performance with multiple clients in parallel

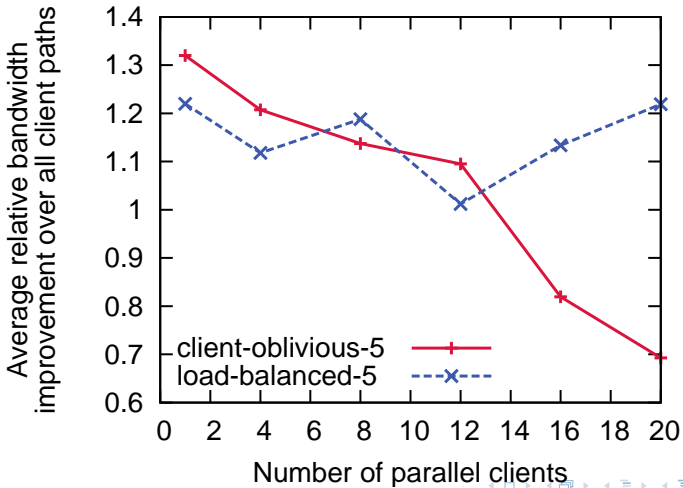
Detouring performance



Public target detouring performance



Load balancing performance



Conclusions

- ▶ **Detouring significantly improves bandwidth on the majority of paths, whilst incurring a fixed detour discovery cost in measurement overhead**
- ▶ Detouring performance is significantly better using public webservers as targets
 - ▶ **Median improvement of 80%**
 - ▶ May suggest a bottleneck in our webserver test platform
- ▶ Detours misprediction does not significantly impact performance on most paths
- ▶ Load-balanced approach retains effectiveness with up to 20 simultaneous clients

Deployment

- ▶ Build out platform as transparent HTTP proxy / browser plugin
- ▶ Aiming for large scale, long-lived, deployment
- ▶ Public deployment via PlanetLab resources

Current research

- ▶ Effectively predicting detour set mapping for clients based on path analysis
- ▶ Optimising dynamic path selection to provide less false-positive detours
- ▶ Investigating what sort classes of paths our approach is most/not effective on
 - ▶ Paths to well provisioned infrastructure (cloud providers)
 - ▶ Paths to geolocated sites (CDN hosts)
 - ▶ Paths heavily restricted by access-link bandwidth (DSL/Cable)

Future research

- ▶ TCP level operation
 - ▶ Harder than HTTP since we cannot easily dynamically switch paths
 - ▶ Thus requires accurate detour path prediction
- ▶ Multi-metric support
 - ▶ Already have approach and routing plane implementation for latency/loss optimisation