Embedding **C**loud-**C**entric-**N**etworking in CCN

M. Sifalakis, M. Monti, C. Tschudin
Dept of Mathematics and Computer Science
University of Basel

CCNxCon, Sep 12, 2012

The Cloud, the Content.. and the User

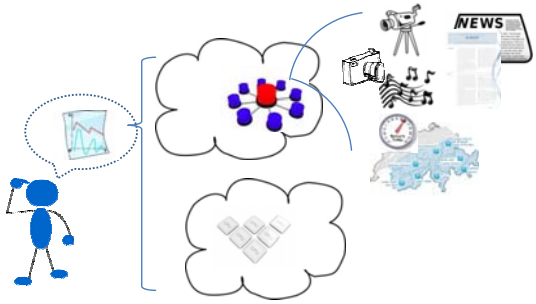
- The Cloud promises ubiquitous resources
  - Storage and Computation
  - On-demand allocation
  - Location transparency

  (all at the edge of the network)

- CCN provides access to content
  - Location transparency (name based)
  - Efficient/Dynamic distribution
  - ... useful for passive content in pre-defined formats ("**cooked**")

  (in the core of the network)

- As Users we wish
  - customized access to content: from **raw**/prime forms to excerpts and digests
  - customized presentation: Resources+capability to create new content (dynamically)



Cloud **C**entric **N**etworking in CCN

- Naming content *and functions*

  By "overloading" the existing API

  Interest (/bring/me (/my/content));

  .. also pipeline the output of one function to the input of another

  Interest (/bring/me (/the/digest (/my/content)));

  .. and using parameters

  Interest (/bring/me (/the/digest (/my/content, "Jan to Sep 2012")));

- CCN as a ubiquitous, native, universal **cloud** technology
  - Extensible set of functions, automatic deployment
  - Custom on-demand content manipulation
  - Caching of results, load-balancing of new computations, in the net

(Example 1)
Starting simple: Revisiting VoCCN

- VoCCN paper: Content source bound/coupled to service protocol

  Interest(/domain/bob/call-id/**rtp**/seq-no);

- Decoupling service function (provider) from content source provider
  - Receiver chooses flavour of transport service

  Interest(/ietf/rtp (/domain/bob/call-id/seq-no);

- Variables (validating *thunks*) for decoupling service look-up from future activation of service (handle or generate content).

  1. Create *thunk* now (and perform lookup)

     Name Transport = **Interest**(/ietf/rtp);

  2. Activate service later

     Interest($Transport (/domain/bob/call-id/seq-no);

(Example 2)
Customise on access, content xcoding

Conditionals

- Testing network conditions

```
If ( $connection in DSL ) then
  NamedFunc Xcoder = Interest(/online/codec/highdef);
else
  NamedFunc Xcoder = Interest(/online/codec/default);

Interest( /bring/me ($Xcoder (/wished/video)));
```

- Checking the service availability

```
NamedFunc Xcoder = Interest(/online/codec/highdef);

If ( ! $Xcoder ) then
  Xcoder = Interest(/online/codec/default);

Interest(/bring/me ($Xcoder (/wished/video)));
```

- Automatically looking for the best possible quality

```
If ( strstr($Content_name, “mpeg”) ) then
  NamedFunc Xcoder = Interest(/online/codec/highdef);
else
  NamedFunc Xcoder = Interest(/online/codec/default);

Interest( /bring/me ($Xcoder ($Content_name)));
```


(Example 3)
Handling content as time-series

- In accessing dynamic content, often need to identify
  - parts of the corpus (time frames), since it may be infinite
  - branches of the corpus (different subsets/evolution paths)
  - versions of content (snapshots = static content)
  - ... and wish-list content (future produced...or derived) !!
- loops and filter conditions

```
for ( Year in range[2001 .. now] )
  /my/cloud/live_albums = Interest(/my/cloud/make/slideshow (/bob’s/photos/$Year));

Interest(/google/picasa/playback (/my/cloud/live_albums));
```

- *Raison d'être* for time-stamping content rather than just sequencing

(Example 4)
Sharing my data

(...while respecting protected IPRs and licensing)

- Publishing virtual (on-the-fly) content: scripting functions

Create live content on demand through (partially evaluated) named functions

```
let analysis_results (Condition...) :==
 Analysis = /my/cloud/famous/analysis/method; // Public licence
 Content = /ipr/protected/dynamic_content;    // Private IPR

 for ( Filt in list [$Condition] )
  /my/or/your/space/$Filt = Interest($Analysis ($Content, $Filt));
  Interest(/inkscape/makesvg (/my/or/your/space/$Filt));

Register(analysis_results, /my/cloud/analysis_results);
```

When content not available or if condition not valid yet, the service "pipe" is still established. If Interest not expired, content delivered as soon as it is available

```
Interest(/my/cloud/analysis_results, cond1|cond2);
```

Static conditions

```
Interest(/my/cloud/analysis_results, /func/genlist/last10yrs);
```

Dynamic conditions (content) provided by named function

Sounds familiar so far? It starts with "λ"

What we need is

- **Names** for abbreviating and publishing (content as well as functions on content)
- Constants
  - True/False on name presence
  - Conditional test: if-then-else
  - Fix point combinator: loops
  - Iterators on content (timestamps more powerful than seq numbers)
  - Operators: Interest(.), Register(.)
- Minimal type-system
  - 1 basic type: content
  - Function types stem their definitions on λ-terms
  - If well-typed we can set rules for distinguishing ops on types of content
  - NOTE: Any function with >1 typed arguments can be expressed as a sequence of 1-argument functions can maps directly to the Interest(.), Register(.) API
- A function formation and function application (reduce) capability:
  - An interpreter in The Cloud

... And we have a typed λ-calculus for content centric networking

Our vision

From single source with multiple users …

… to multiple sources (content and functions) and multiple users (choosing any combination)

http://named-function.net/

(Additional discussion points #1) Naming functions/code

- Content manipluation so far only through application level facilities, executed at the edge → in-network proxies
  - For the user (e.g. Mobility)
  - For the service (e.g. Event exchange, indirection)
  - For the content (E.g. “Late binding” for dynamic content)
- Caching functions/code: **Moving code around**
  - increase service availability and re-use
  - reduce latency, distribute processing load
  - routing: redirectors
  - effective introspection of network state (since it will be cached along the path between the user and the content source)
  - Controlled case: Can be used to update the network fabric opportunistically (e.g. deploy new protocols, software updates, SDNs)

(Additional discussion points #2) Named functions & multiple transports

- On-demand deployment of different transport strategies for different types of content
  - Transport layer closer to the application
  - User selects from a number of options
  - Content provider can offer the options
- Decouple congestion avoidance/remediation logic (in-net) from flow/rate control actions (receiver/client end)
- Small extension, consistent with the specification/conventions
  - Enabler for the strategy layer
  - CCN thin layer remains simple and elegant