

# Named Functions ( $\lambda$ -reduction inside CCN)

[www.named-function.net](http://www.named-function.net)

The  **$\lambda$ -calculus** is a formal system for *name binding* and substitution – it is the root of all functional programming languages (LISP, Haskell etc). From  $\lambda$ -calculus's perspective, CCNx is a protocol to do name resolution i.e., to do a variable lookup. **In Named Function Networking, we extend CCN to reduce all three forms of  $\lambda$ -terms:**

E := a variable

E := f(e) application (of function f)

E :=  $\lambda x.e$  abstraction (x is the param)

## NFN-Example 1: Request a transcoded video

- needs two names (video, and transcoder)!

[ ccnx:nfn | /name/of/data | /name/of/transcoder ]

## NFN-Example 2: Replace CCNx' implicit hash

- with CCNx, a client can filter on the content's digest  
- write this as a program:

```
define filter(dataName, hashVal) (
  (ifelse (eq (sha256 dataName) hashVal)
    dataName
    nil)
)
```

NFN resolver's task is to find suitable execution site

## How to turn CCN into a $\lambda$ -term resolver:

- NFN names are  $\lambda$ -expressions
- NFN first checks for *cached computation* results (using the „find-or-execute“ instruction FOX that searches for a result bound to the hash of the term to resolve).
- If no cached result is available, NFN reduces the term (using “Krivine's lazy abstract machine”) and proceeds with sub-terms etc until we have a variable lookup or a function execution.



**CCN-lite** is a lightweight implementation of the CCNx protocol. It supports most of the essential CCNx functionalities, and more:

- *Tiny code base*: The core CCNx logic keeps in less than 1000 LoC
- *Identical code* for three incarnations: Linux kernel, user space, OMNeT++ simulator
- *Scheduler support*: both at chunk and packet level
- *Fragmentation*: CCNx over Ethernet
- *Management*: via CCNx msgs
- builtin, small *HTTP server* for quick diagnostics
- *ISC licence (BSD-style)*
- Finally: *interoperable* with CCNx !

### Ideal for:

- class room work
- experimental extensions
- non-caching relays
- code base for commercial products

### Status:

- code is on GitHub
- release 0.1.0 in July 2013
- used by Cisco, Freie Uni Berlin (RIOT), U of Basel

Modules that can be selected at compile time:

#defines:

*USE\_CCNXDIGEST, USE\_DEBUG, USE\_DEBUG\_MALLOC,*  
*USE\_FRAG, USE\_ETHERNET, USE\_HTTP\_STATUS,*  
*USE\_MGMT, USE\_SCHEDULER, USE\_UNIXSOCKET*

*Support for NFN to be added soon!*

