

The Name of the Networking Game ...

... is Term Rewriting

ICNRG interim, Paris, Sep 27, 2014

```
# This is a m4 program for networking people:  
define(y, `Y()me')dnl  
define(X, `g''')dnl  
define(x, `n''')dnl  
define(Y, `a')dnl
```

The $x()y$ of the $X()y$ is $x()y$ rewriting.



Christian Tschudin, University of Basel

What Networking is all about; What this talk is all about

```
tschudin@talo: ~/pap/2014-09-icnrg
% m4 <whatnetworkingisallabout.m4
# This is a m4 program for networking people:
The name of the game is name rewriting.
% █
```

This talk is:

- an attempt to **draw a map** of ICN approaches, including named fcts
- not about λ -Calculus

Name Rewriting is Daily Networking Business

parc.com



omega.xerox.com



13.1.64.95



...

(MAC address, VLAN port, SDN magic ...)

Networking Protocols as **Term** Rewriting

Request/reply protocols (in a LAN), e.g. `ping alto.xerox.com`

`<nodeA, src_ip, dest_name, request>`

| DNS

`<nodeA, src_ip, dest_ip, request>`

| ARP

`<nodeA, src_ip, dest_eth, request>`

| xfer

`<nodeB, src_ip, request>`

| ICMP

`<nodeB, src_ip, reply>`

| ARP, xfer

`<nodeA, reply>`

Rewriting Systems

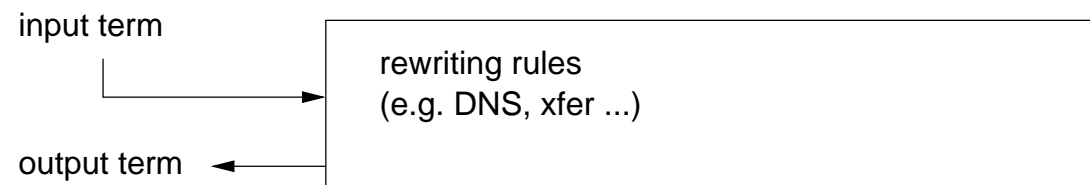
Start with term — apply “rewriting rules” — start over

(as does m4)

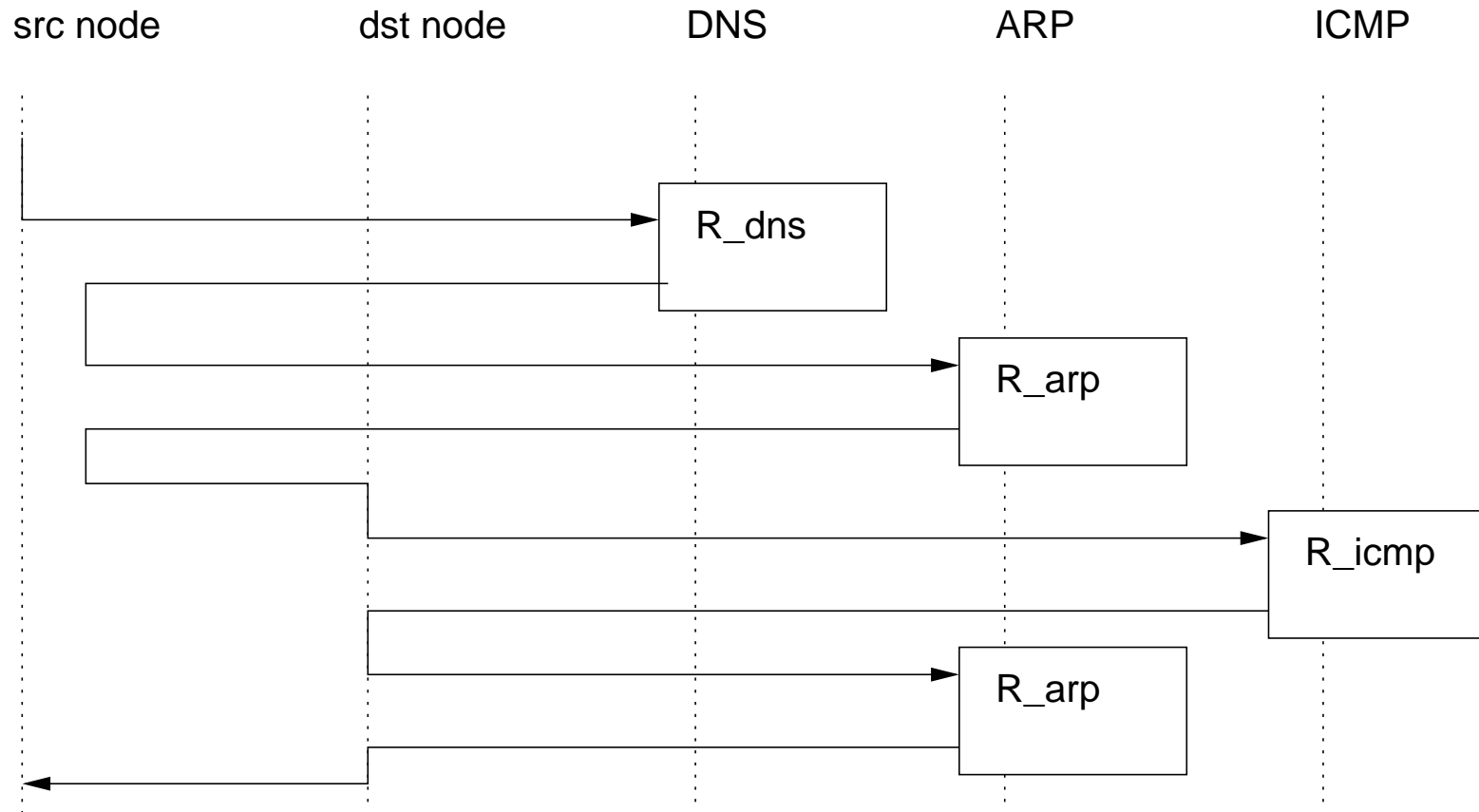
∃ Turing-complete rewriting systems, e.g. PCP

A graphic notation for rewriting rules in networking:

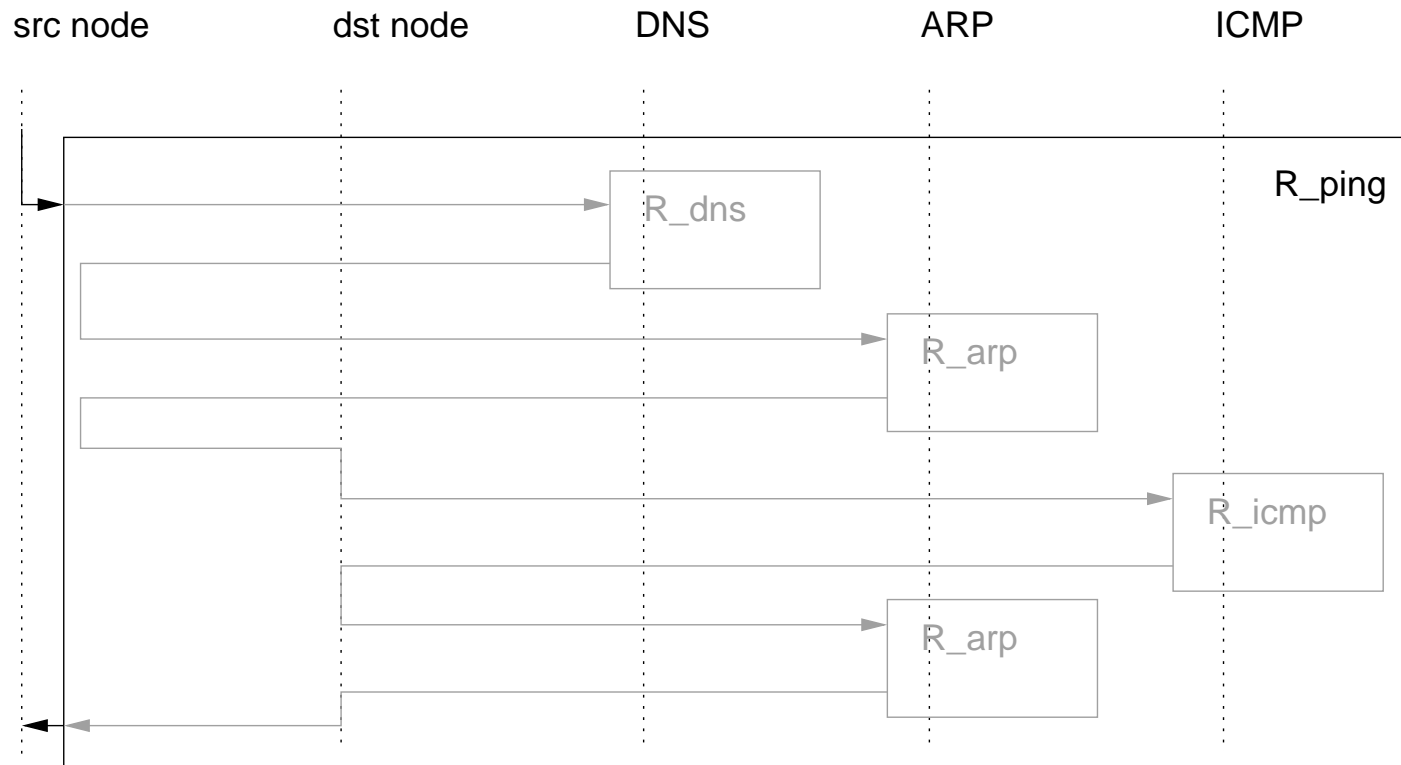
- DNS (a whole DB with replacement rules)
- ARP (substitution)
- transfer rules (bit shipping, including an alternate rule called “drop”)
etc.



Term Rewriting for Networking Protocols (graphic)



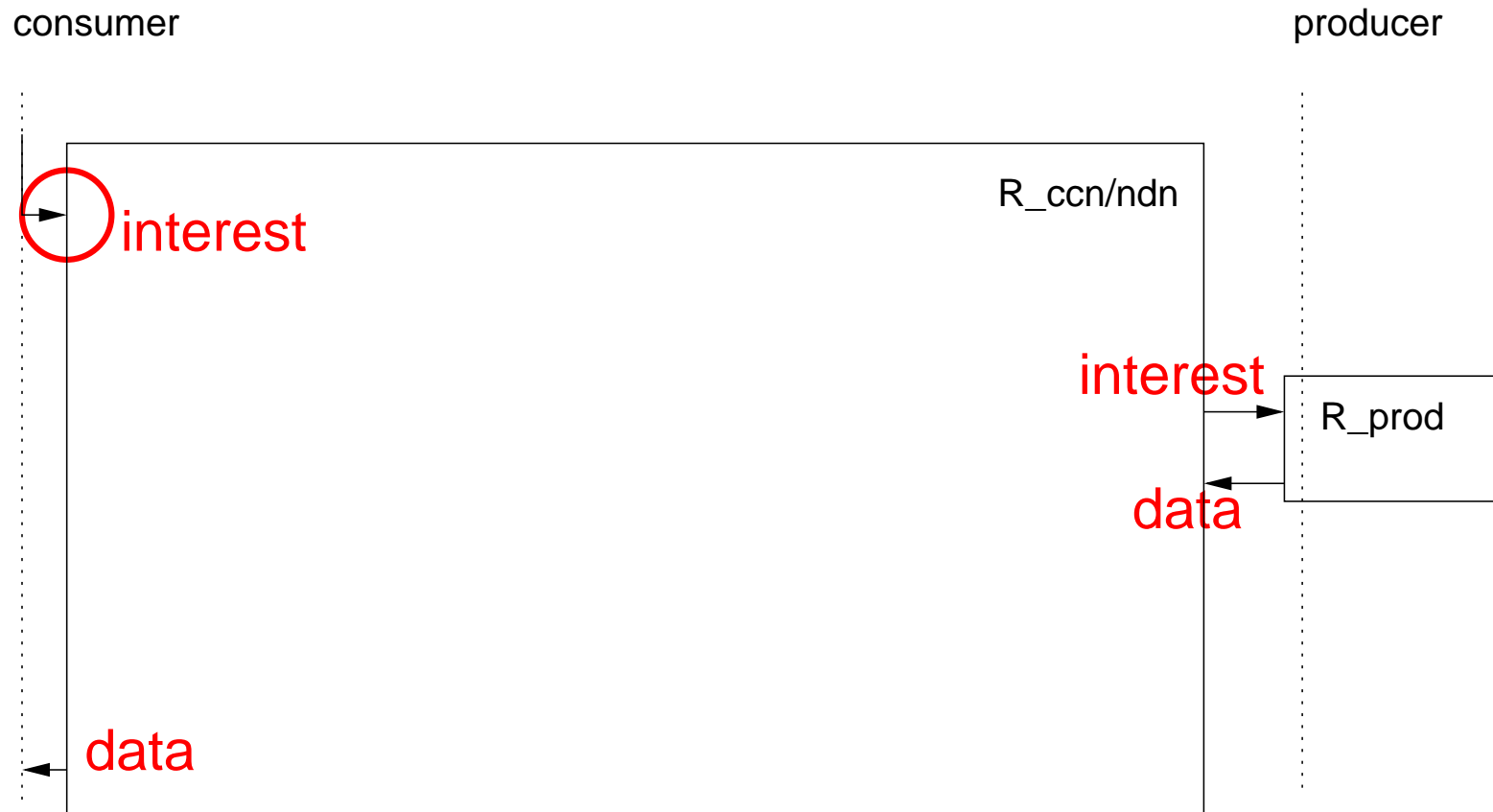
PING as a rewriting box – Hey, it's distributed computing!



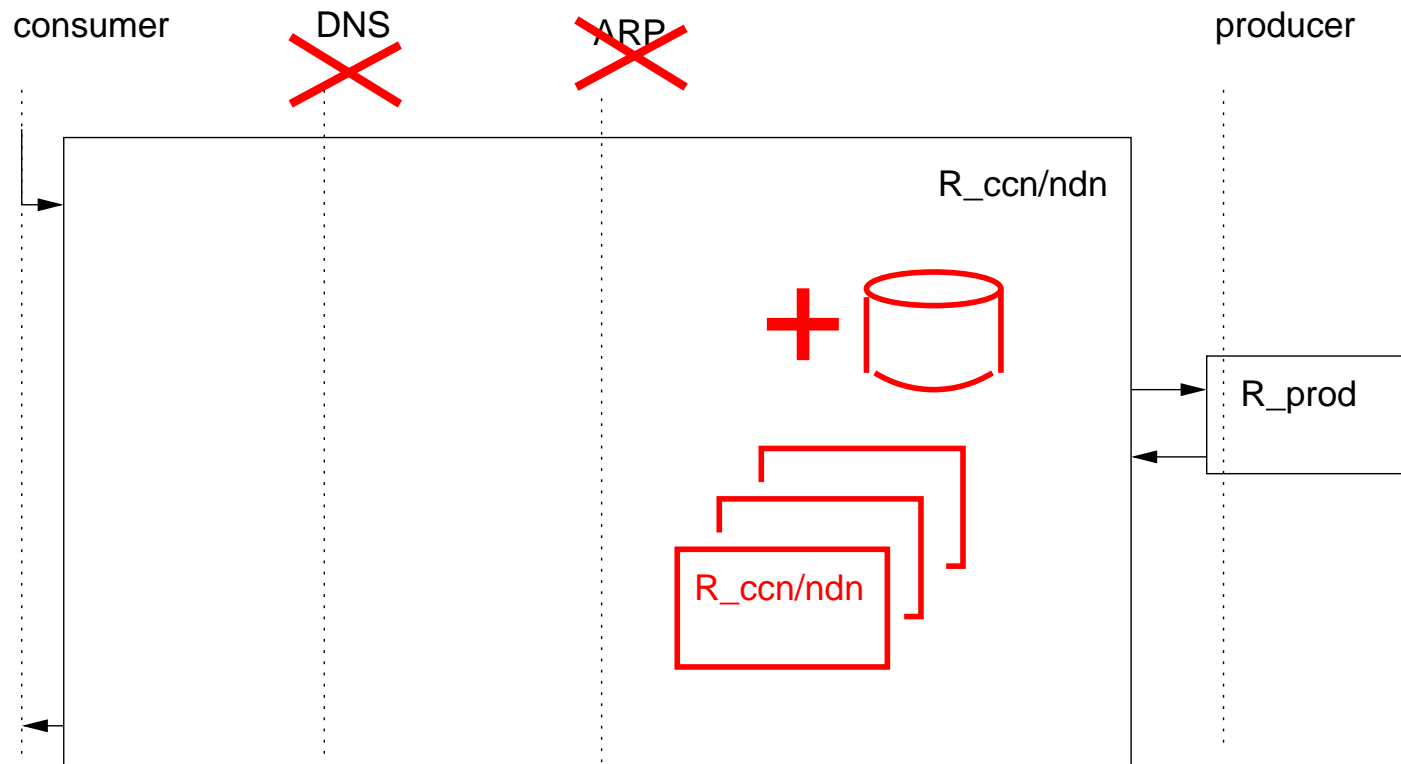
Potentially could capture (abstract, compose) all networking that way.

But: \exists ugly sub-boxes (routing), little theory progress

CCN/NDN as Rewriting



Important contribution of CCN/NDN: self-similarity

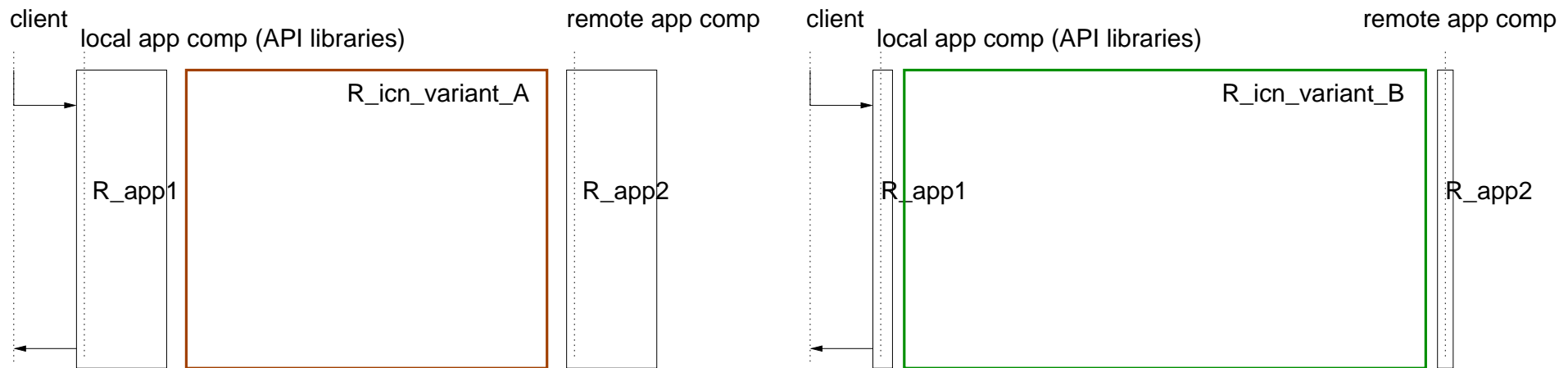


Usually mentioned first: in-network memory, no addresses, no ARP, etc

New (and important IMHO): unify service access and peering protocol
(~ recursive boxes, “the-new-waist-as-a-fixpoint”)

Network Rewriting Boxes and Applications

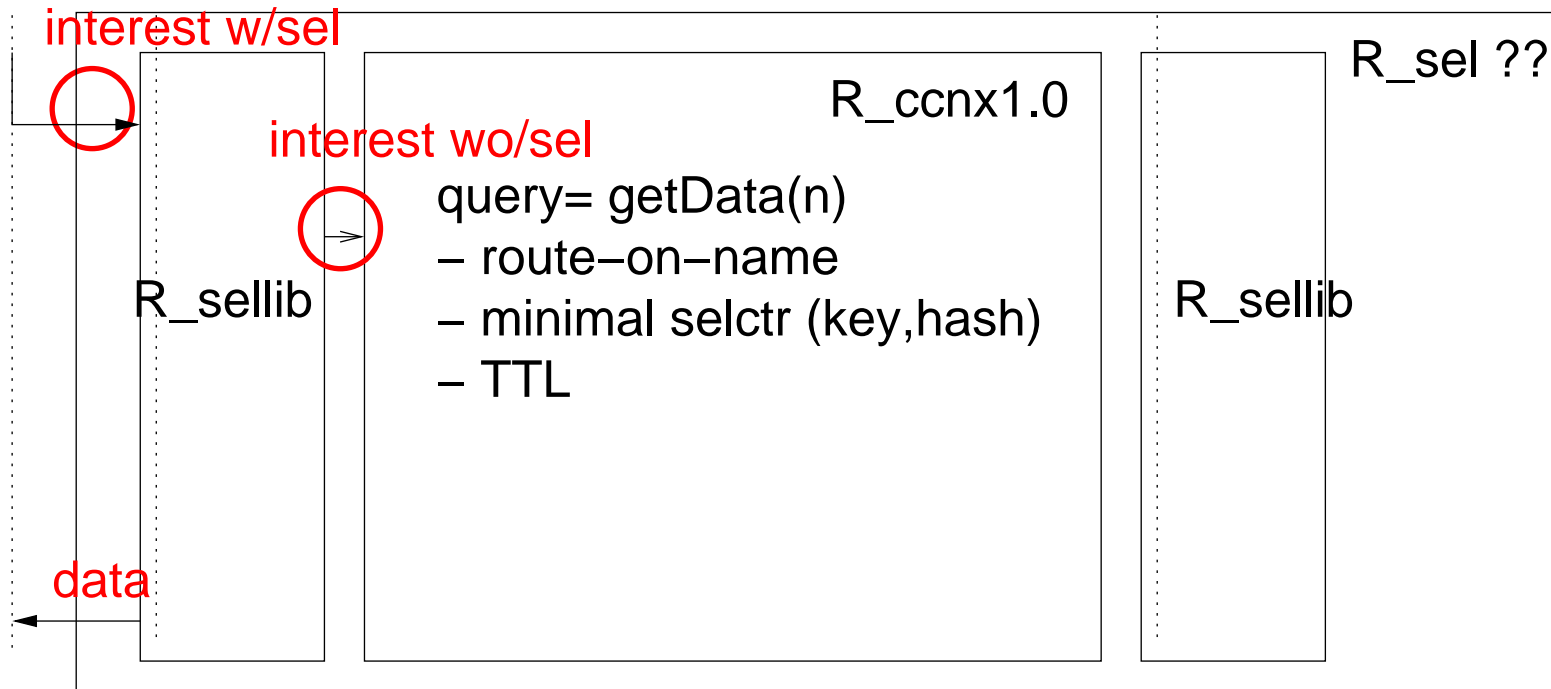
Service creation: composing application logic and an ICN networks
(whose **expressiveness** might vary - this is our debate)



Spectrum from pure-transport to network-does-it-all,
and in different ways: → **gallery**

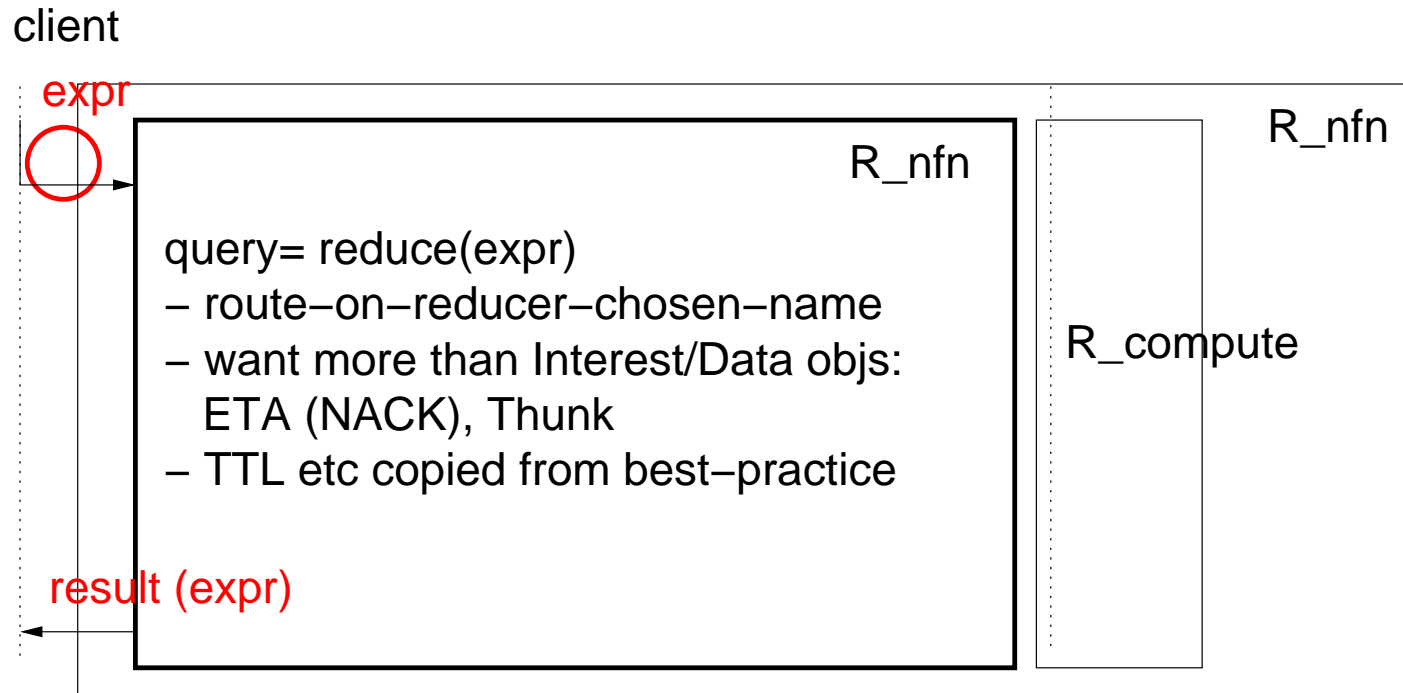
Gallery: CCNx 1.0 and NDN

client



Bold claim (=non-trivial): $R_{CCNx1} + R_{SelectorLib}$ is equivalent to NDN.

Gallery: Lambda-Calculus (NFN)

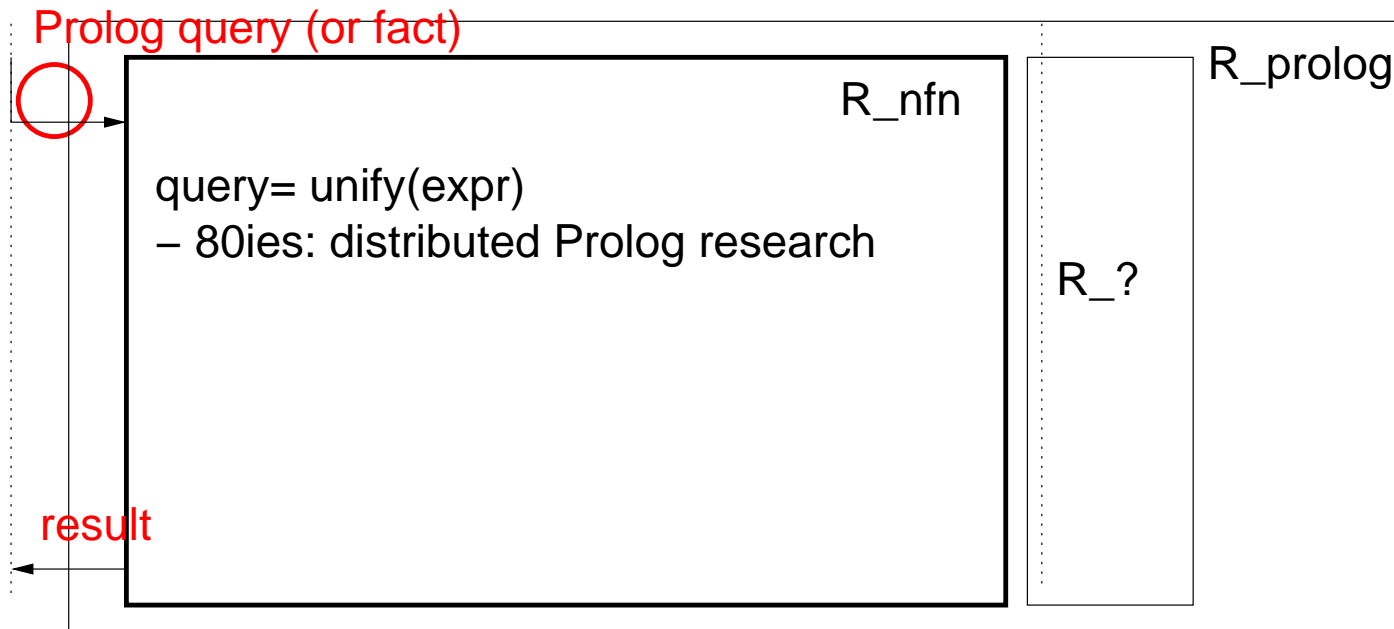


Non-trivial claims: R_{nfn} can emulate all ICNs; NFN is a fixpoint

Emulation argument: define `getFromNDN(NDNname)`, `getFromCCNx(CCNname)`, plus define name mangeling to map name spaces

Gallery: Prolog? – rewriting based on facts and rules

client



Powerful “unification” concept: $\text{dns}(\text{name}, X)$,
but also $\text{dns}(Y, 10.11.12.13)$

Restricted (finite) form of Prolog: **Datalog** (\rightarrow NDlog papers, 2006-09)

Gallery: Service Chaining

see ACM ICN 2014 conf

M. Arumathurai et al: *Exploiting ICN for flexible Management in Software-Defined Networks* (best paper award)

The “box” offers hop-by-hop rewriting of names

T. Braun: Service centric networking

Gallery: Database query languages

SQL, recursion, and (builtin) functions

Gallery: Descriptor based

more to explore

Concluding Observations, Questions

- Counter-intuitive:
 - NDN has **constant name-to-object binding**
 - Named-Function does repeated **rewriting of terms**.

This is a key difference:

named-data **has** implicit term “`getData(name)`”
but this is not exposed, cannot be changed.

- Who will be layered on top of whom?
NDN-over-CCN, NFN-over-NDN, ANY-over-NFN, ...
- How to couple/peer different Named-Function styles
and named-data transport networks?

Named Function Networking Agenda (Paris, 2014-09-27)

1. **Intro** to Named-Function-Networking (Data-on-Demand) (10')
2. **Other views**, projects, prototypes, plans? (15')
NFN for NDN/CCNx, Service Centric Networking, NFN over NetInf?
Cloud interfaces? NFN for IoT? Other? *NDNex, Sneak peek*
What style? imperative vs declarative, Linda, DB semantics . . .
3. **Requirements** for data-only suites (NDN, CCNx) (15')
packet formats (expressions, thunks, results), NACK, ETA response,
PIT-support, route announcements for servers
4. **Security** (10') – secure coupling, trusting results, (data) access filters
5. Future work: SIG or not? (5')