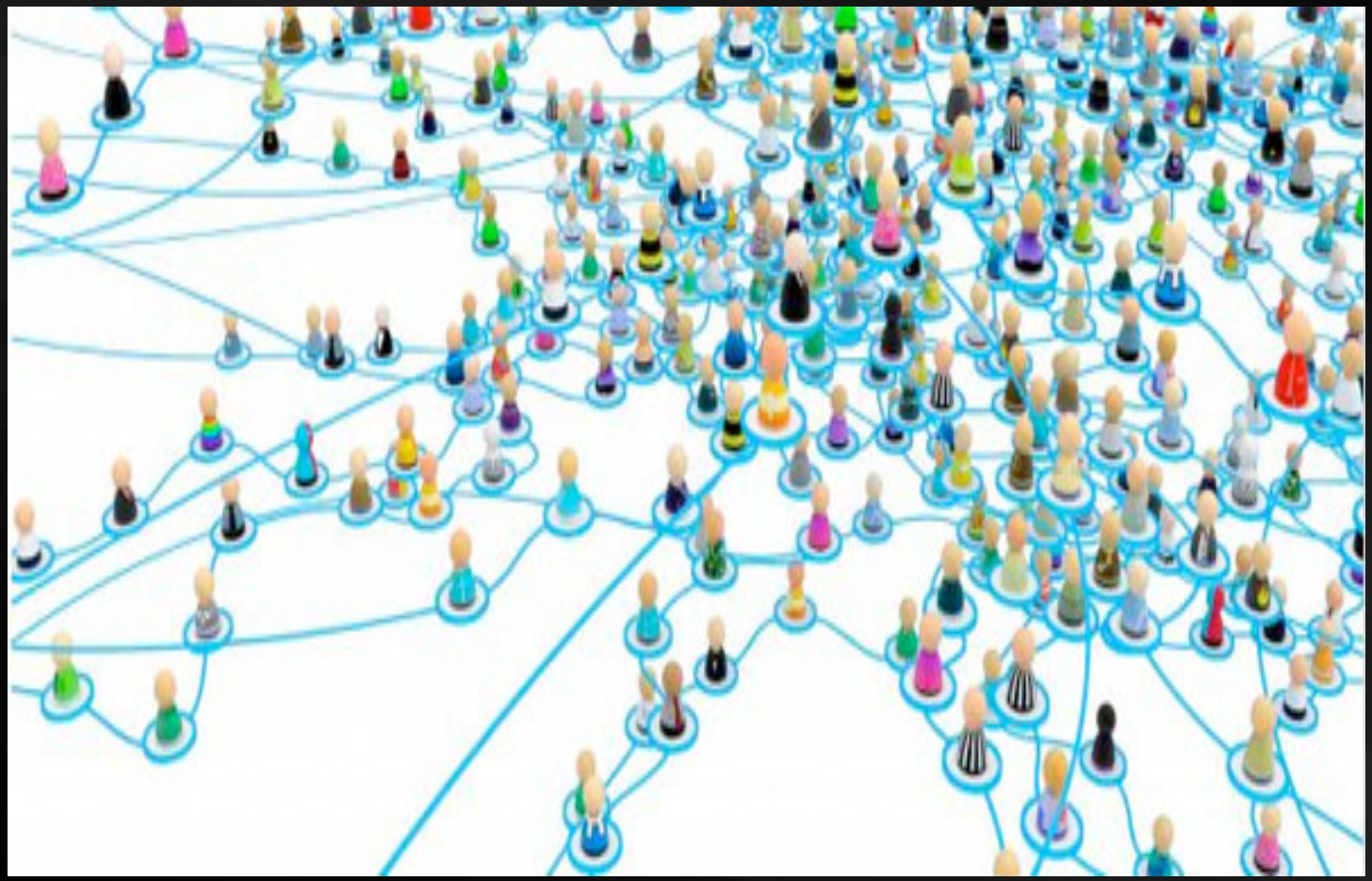
Software Defined Networking, Erlang, and the Future of Computing

Stu Bailey, Founder/CTO

Infoblox
CONTROL YOUR NETWORK

How Can We Program an Internet of Things?



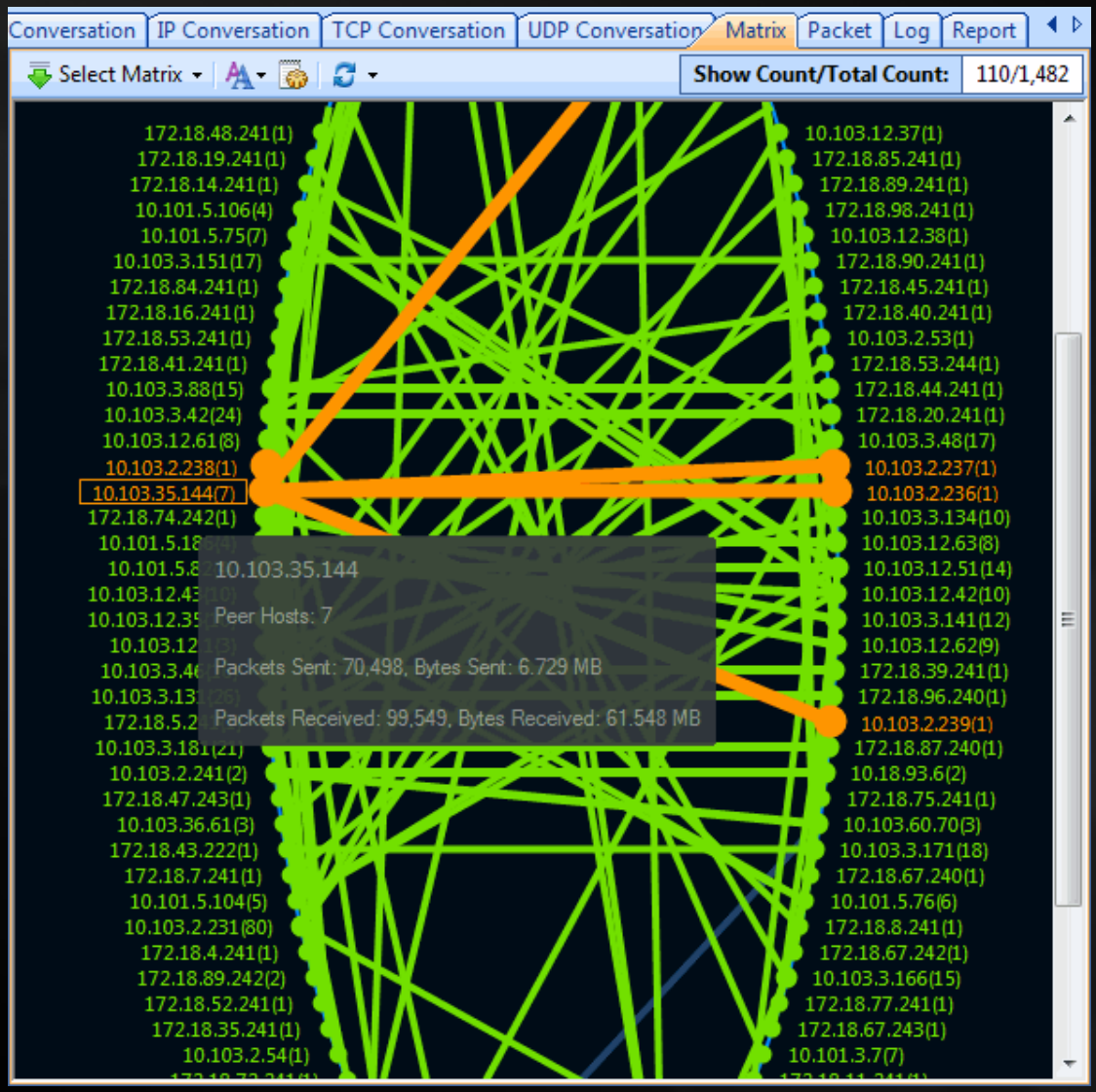
How Can We Program a Million Cores?

And We Must Assume Hardware is Failing (or changing) ALL the Time: “write once run forever”

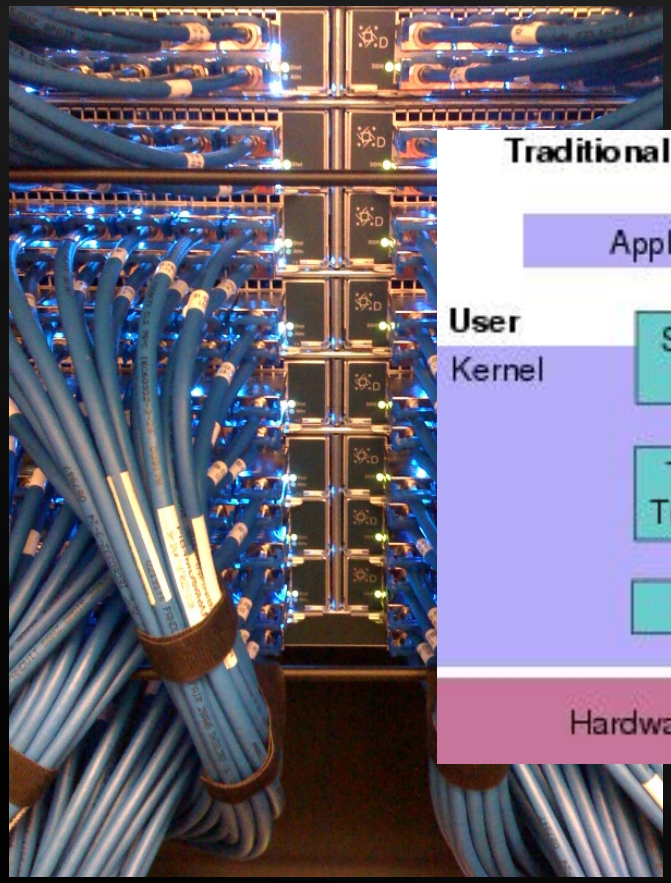
Of Course Erlang!!

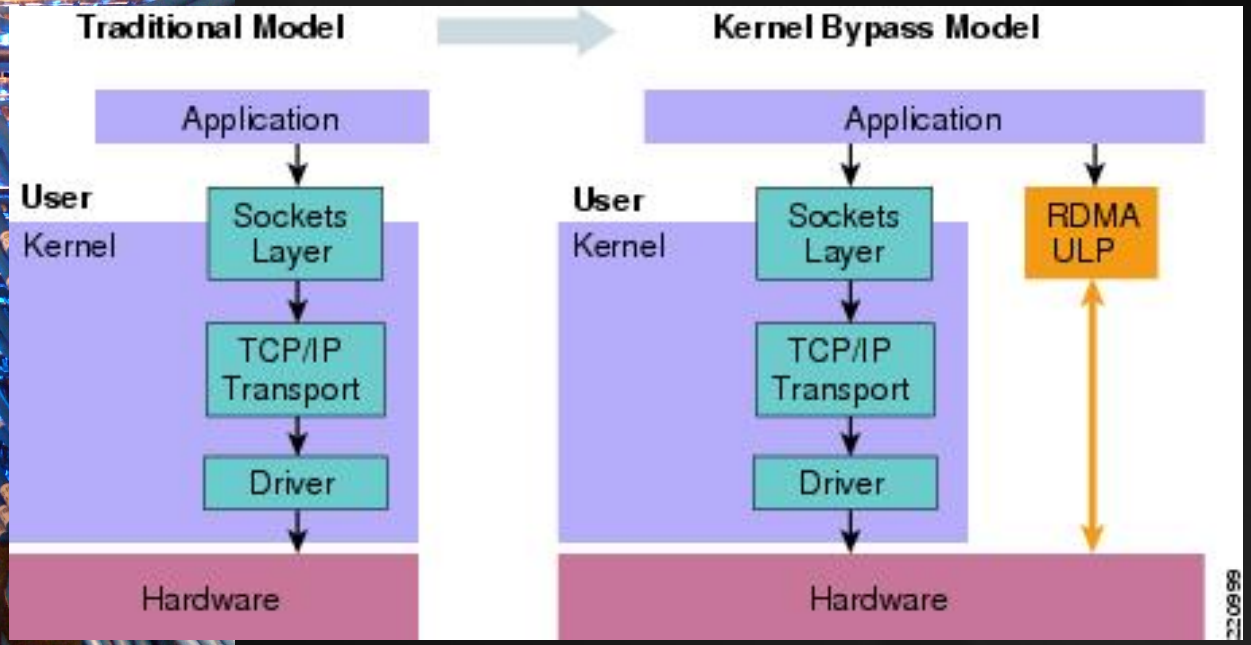
(or Scala/AKKA or node.js)

But wait! What about the network?!

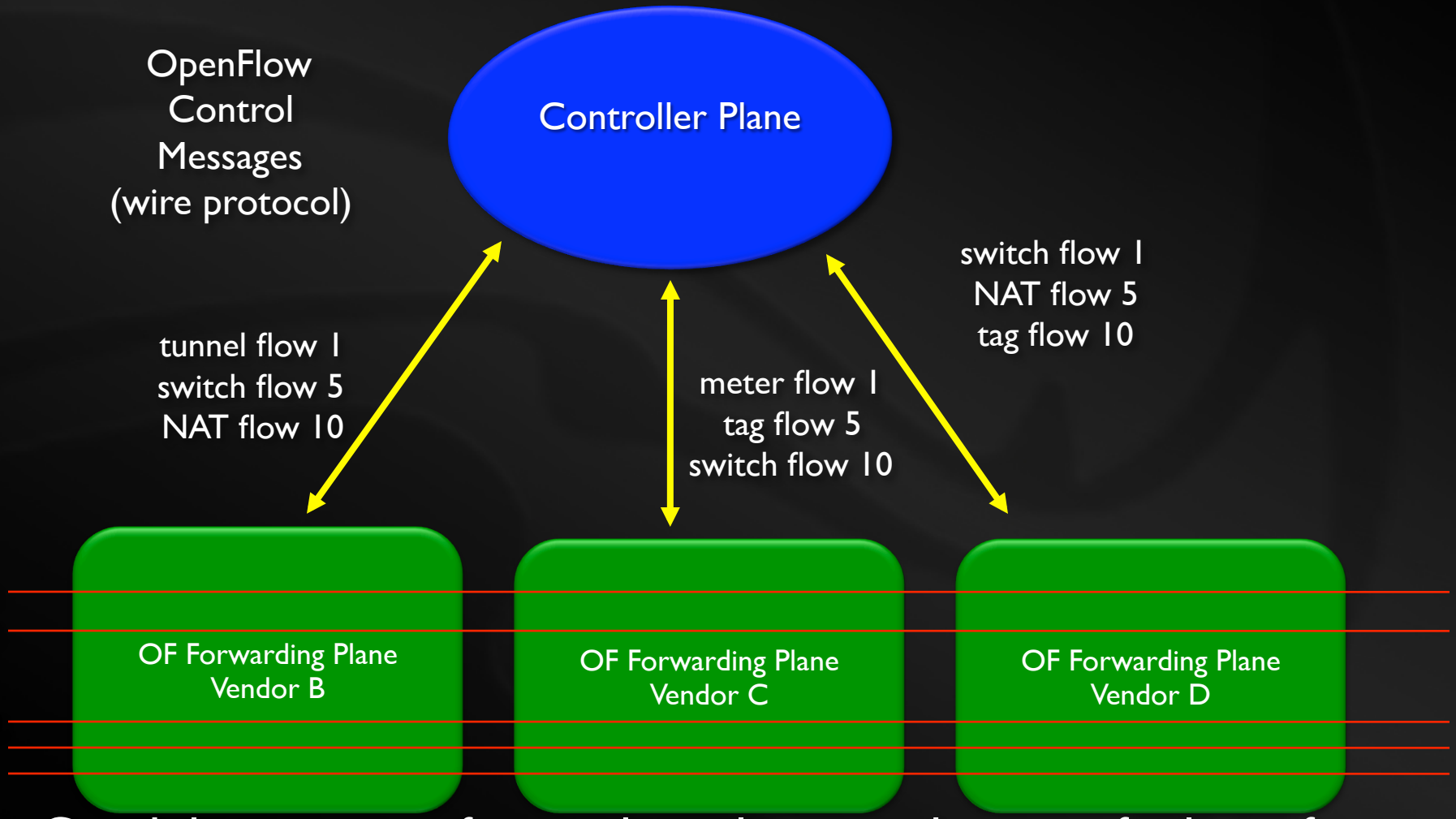


Of course Infiniband! Except…





A STANDARD Forwarding Plane and Logically Centralized Controller Plane



Capabilities across forwarding plane vendors are fairly uniform
Performance and capacity are primary differentiators

FlowForwarding.org

OPEN SOURCE SDN STACK

ENTERPRISE FOCUSED

and maybe more?

Full SDN Stack Timeline

FlowForwarding Community launched (June 2012)
+ alpha LINC Switch code availability

LINC Switch beta v1.0 (Fall 2012)

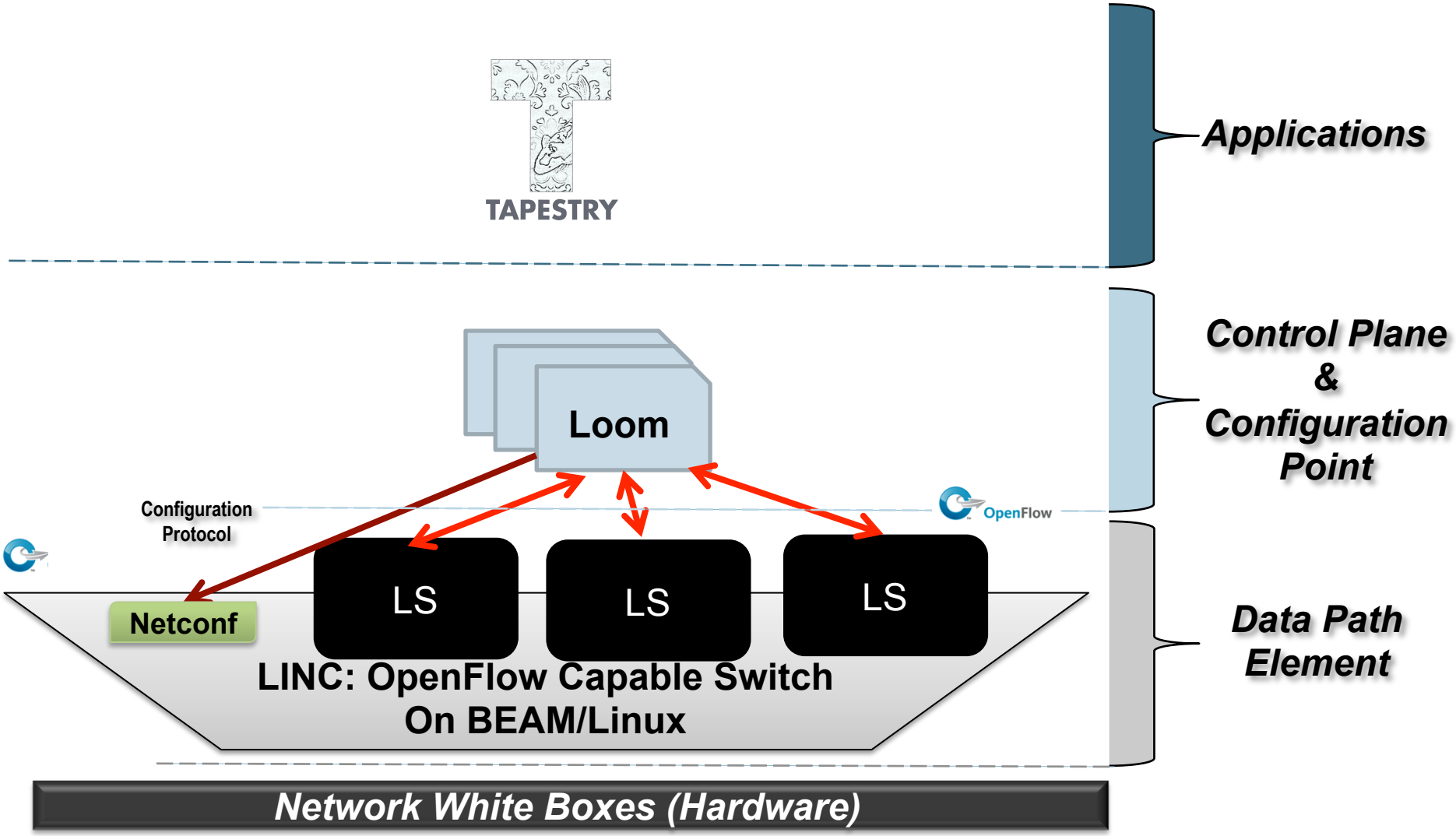
LINC Switch beta v2.X (2013)

Loom Control Plane POC (Oct. 31, 2013)

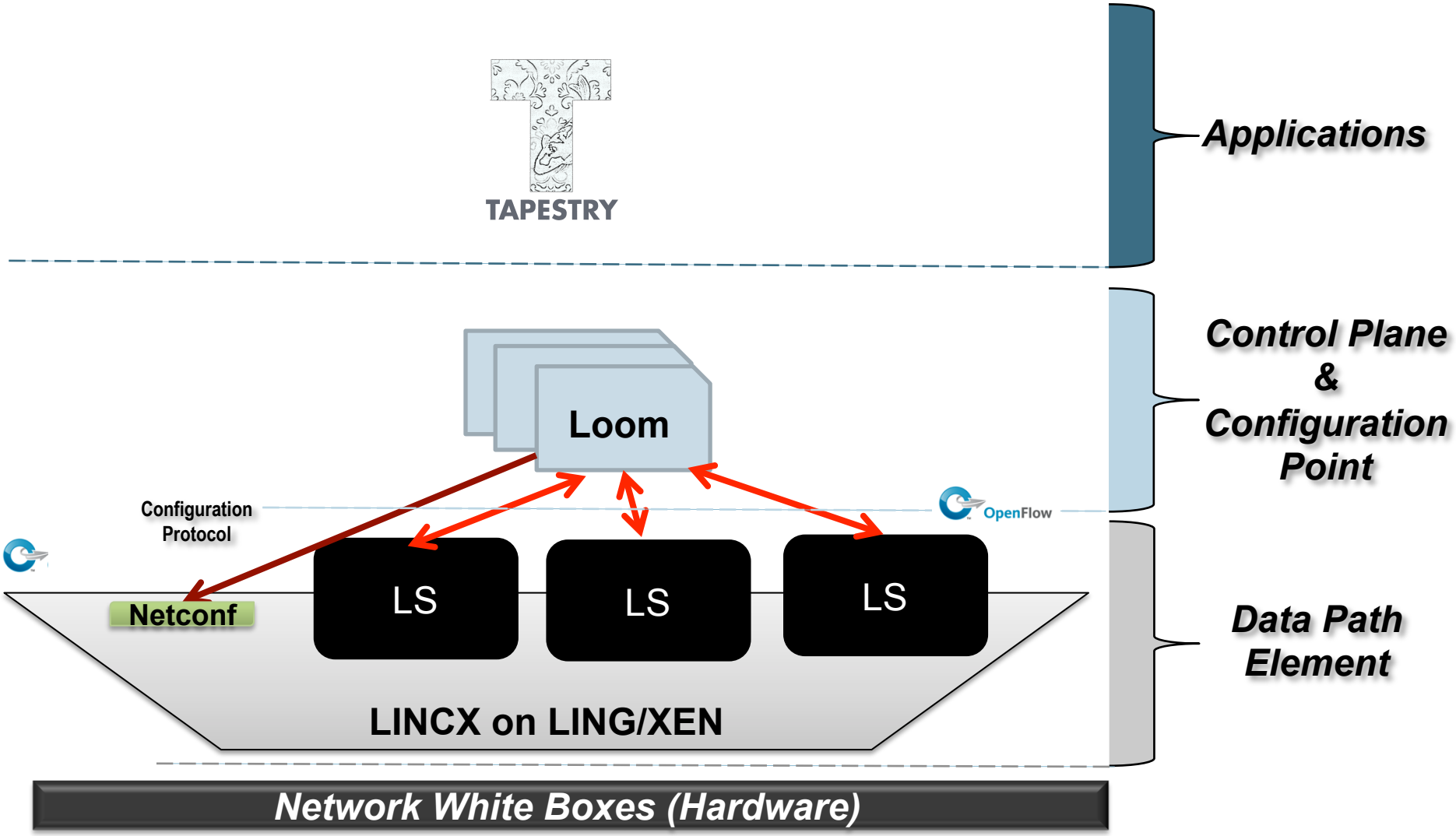
Tapestry Network Complexity Analyzer v0.1 (Oct. 31, 2013)

Loom v2 Scalable SDN Control Plane (Spring 2014)

Erlang = SDN + Big Data

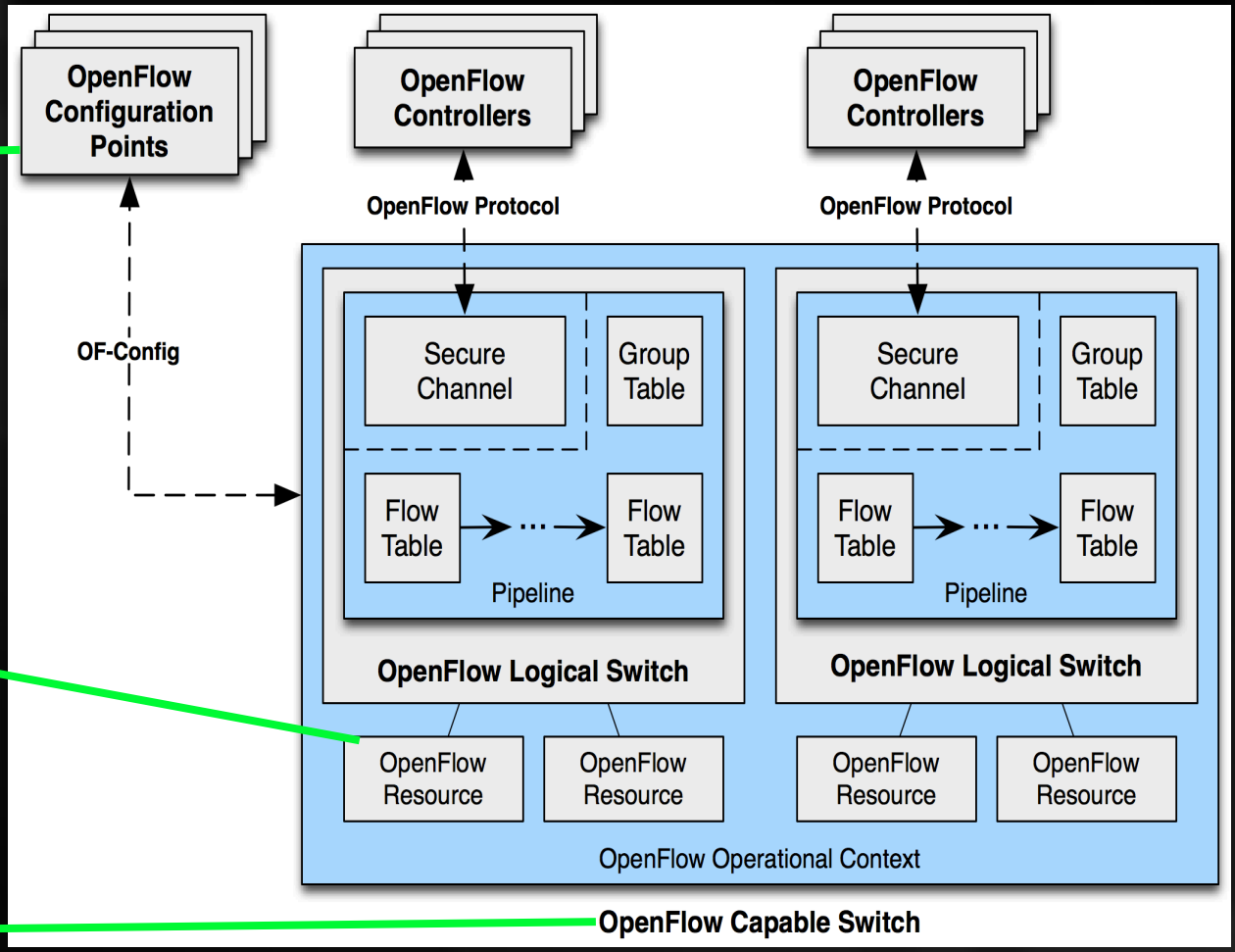


Erlang = SDN + Big Data



LINC Switch Architecture

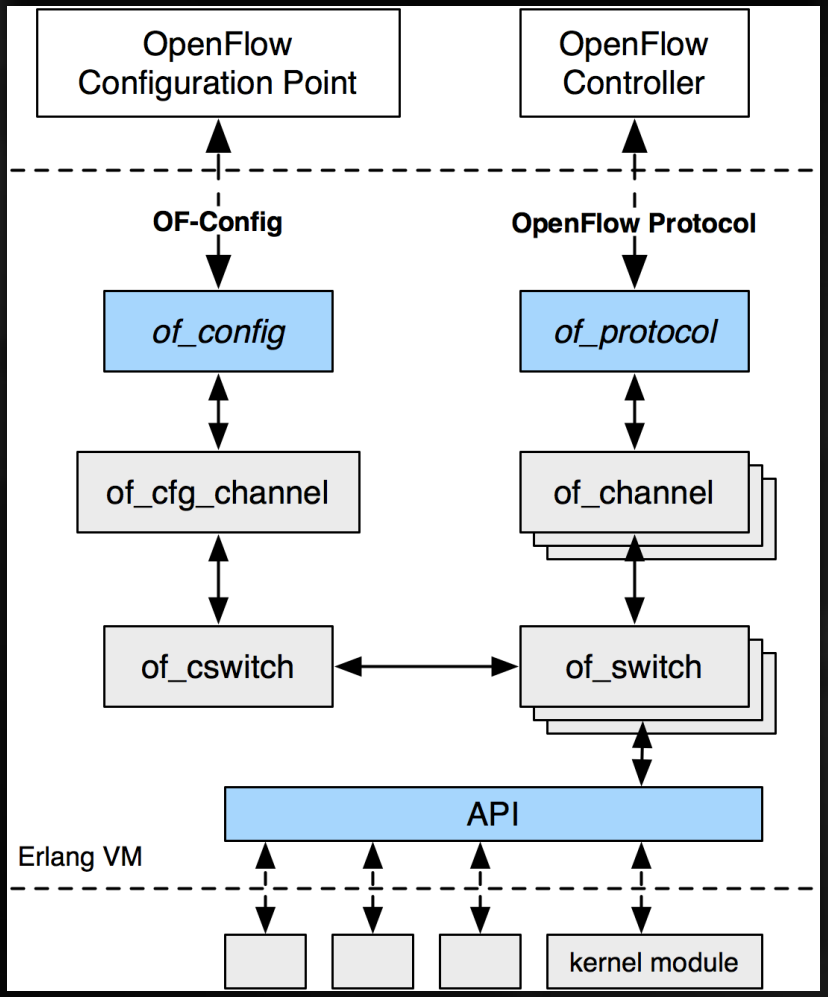
- Provisions OF Capable Switches
- Defines instances of logical switches
- Dispatches resources between switches



- Physical Ports

- Consists of ports & forwarding engine
- Container for multiple logical switches

Erlang Implementation Architecture



LINCX

Current results

3/5/14

Maxim Kharchenko

Cloudozer LLP

Overview

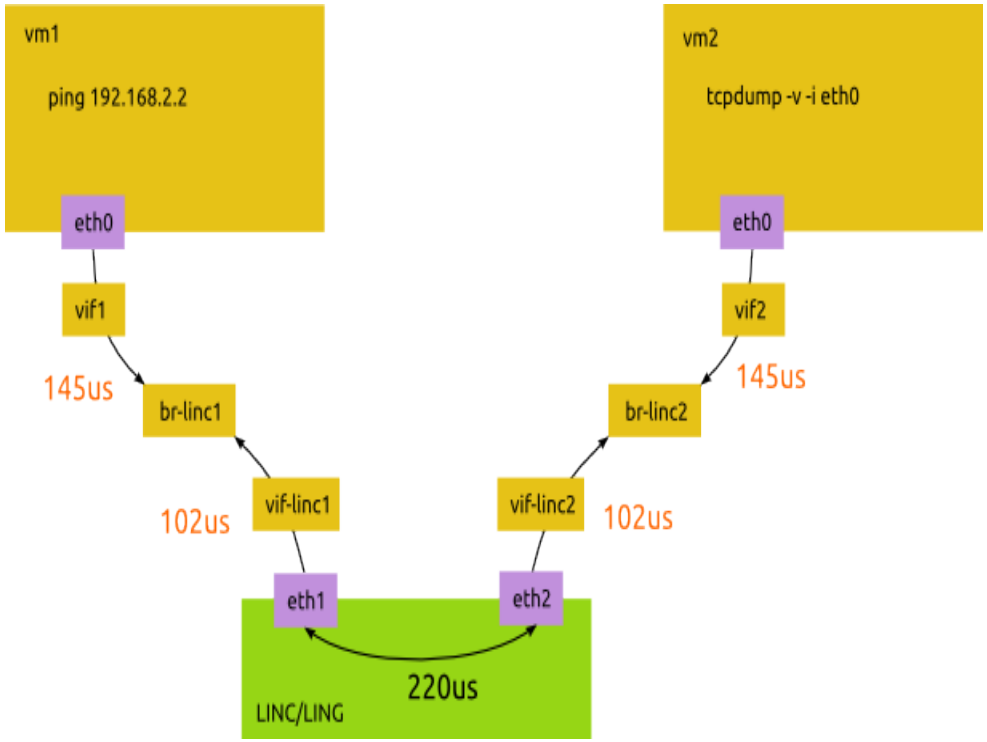
- Started December 9, 2013
- Initial scope = porting LINC to LING VM
- High degree of compatibility demonstrated for LING VM
- Extended scope = fix the LINC fast path
- Beta version of LINCX open-sourced March 3, 2014

Old backend (linc_us4)

- Trivial flows = 2 forwarding rules

	LINC/BEAM	LINC/LING
Max TCP throughput	140Mbit/s	200Mbit/s
Peak memory consumption	4.5GB	98MB
CPU utilization	200%	100%
Processing delay	-	~~220µs~~

LINC/linc_us4/LING latency

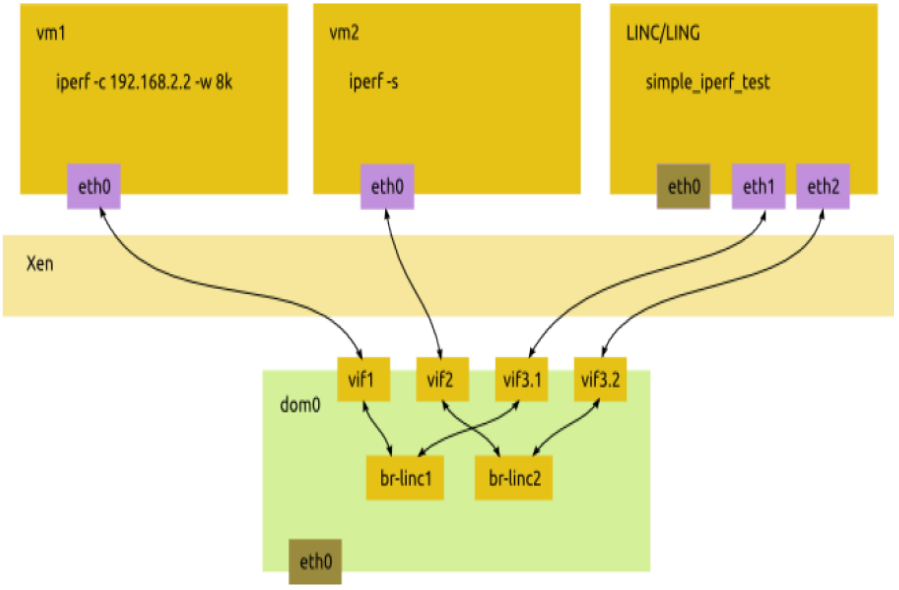


New backend (linc_max)

- Flow table = Erlang module with match() function
- Flow modules regenerated/compiled/reloaded for each flow_mod message
- GC avoided on the fast path
- Packet data (almost) never copied
- Memory footprint minimized

Live demo

- HP Proliant iLO 4 server
- Xen 4.2 hypervisor
- Bridged setup with two VMs:



linc_max 100x faster

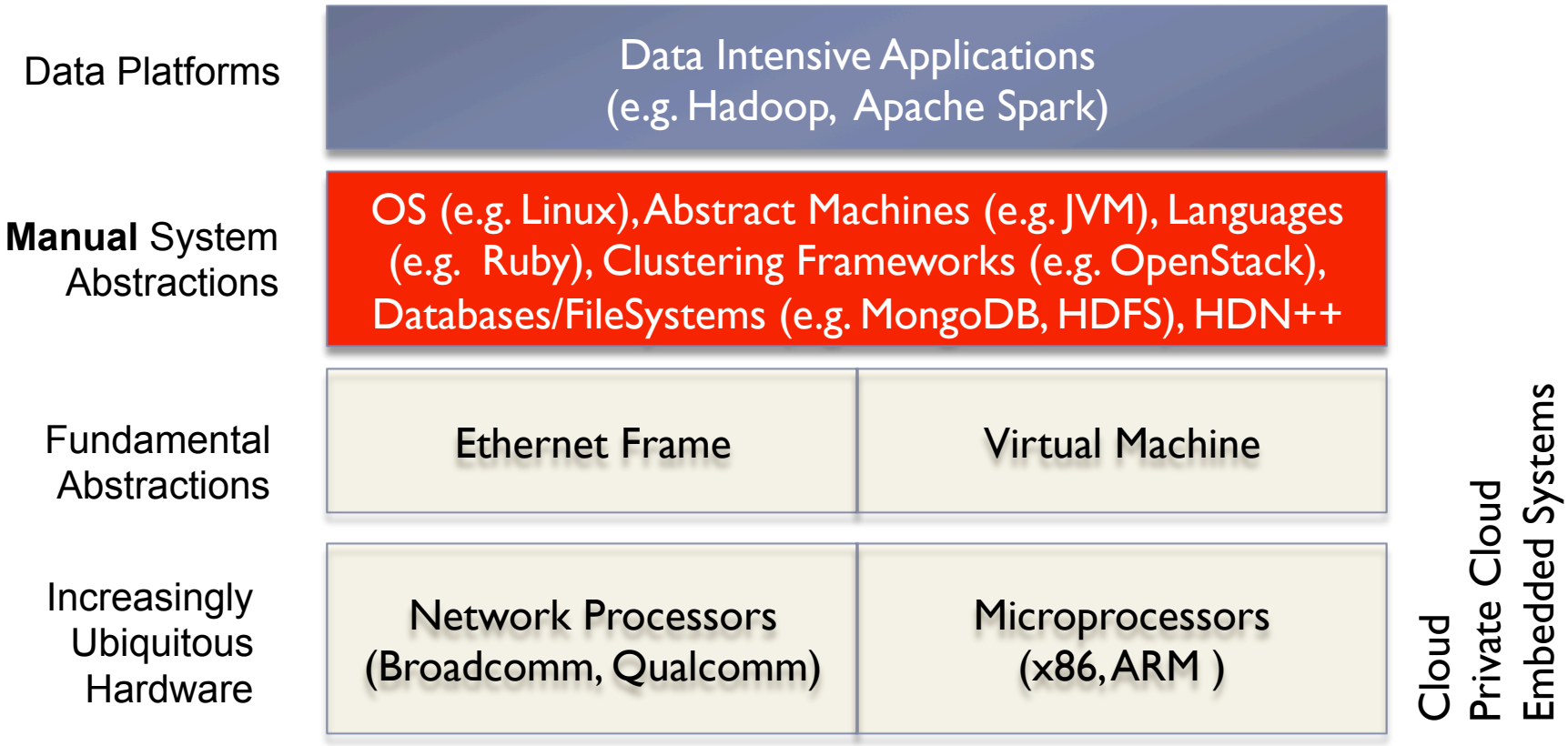
- A non-trivial flow table with 256 rules:

	LINC/linc_max/LING
Max TCP throughput	2.5Gbit/s
Peak memory consumption	22M
Processing delay	3µs

Future

- Multicore
  - A switch per port? Per two ports?
- Faster software bridges
  - Replace generic Linux bridges with faster alternative
- PCI passthrough
  - Eliminate Linux completely from the packet path

The State of the Art...is Not

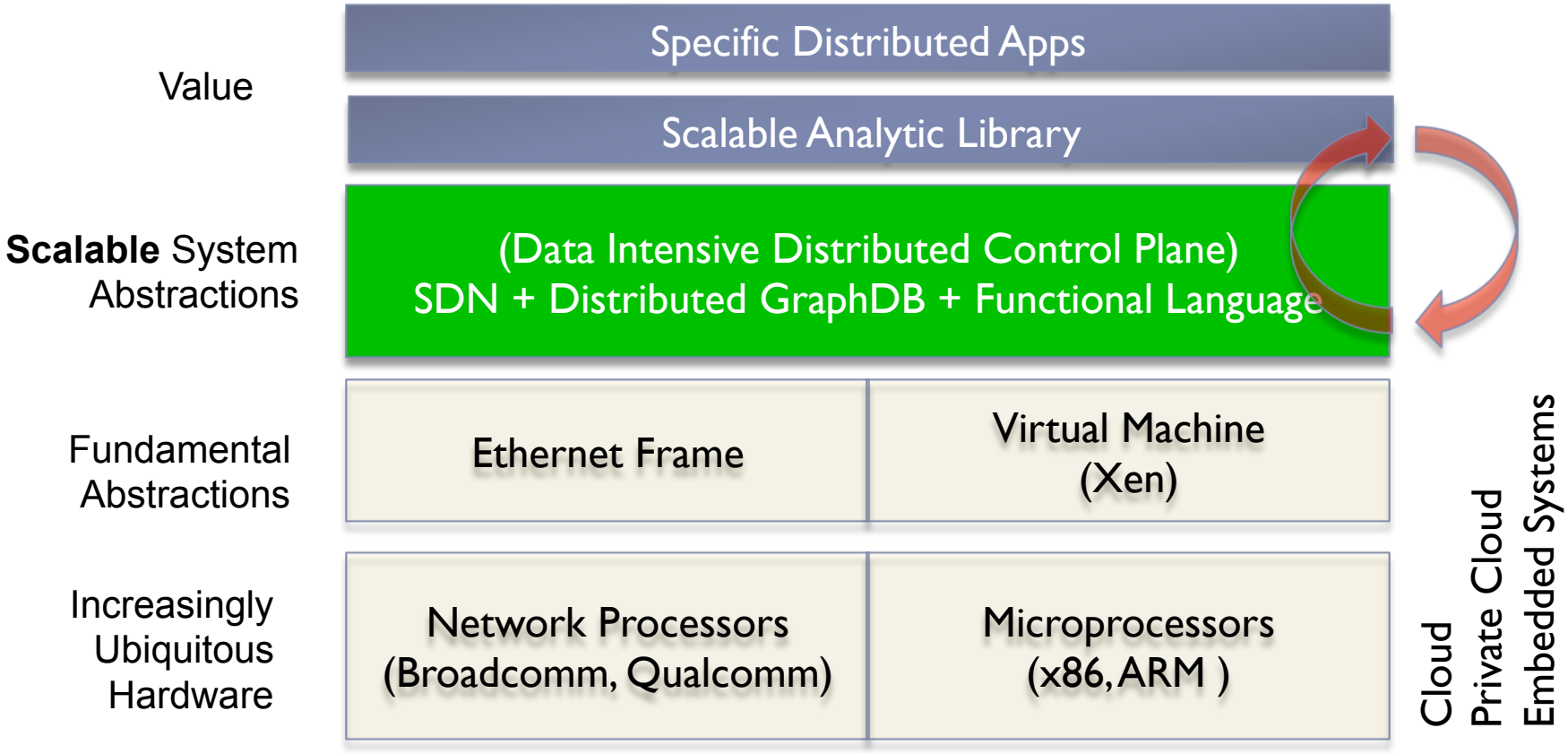


The right perspective…and perfect timing

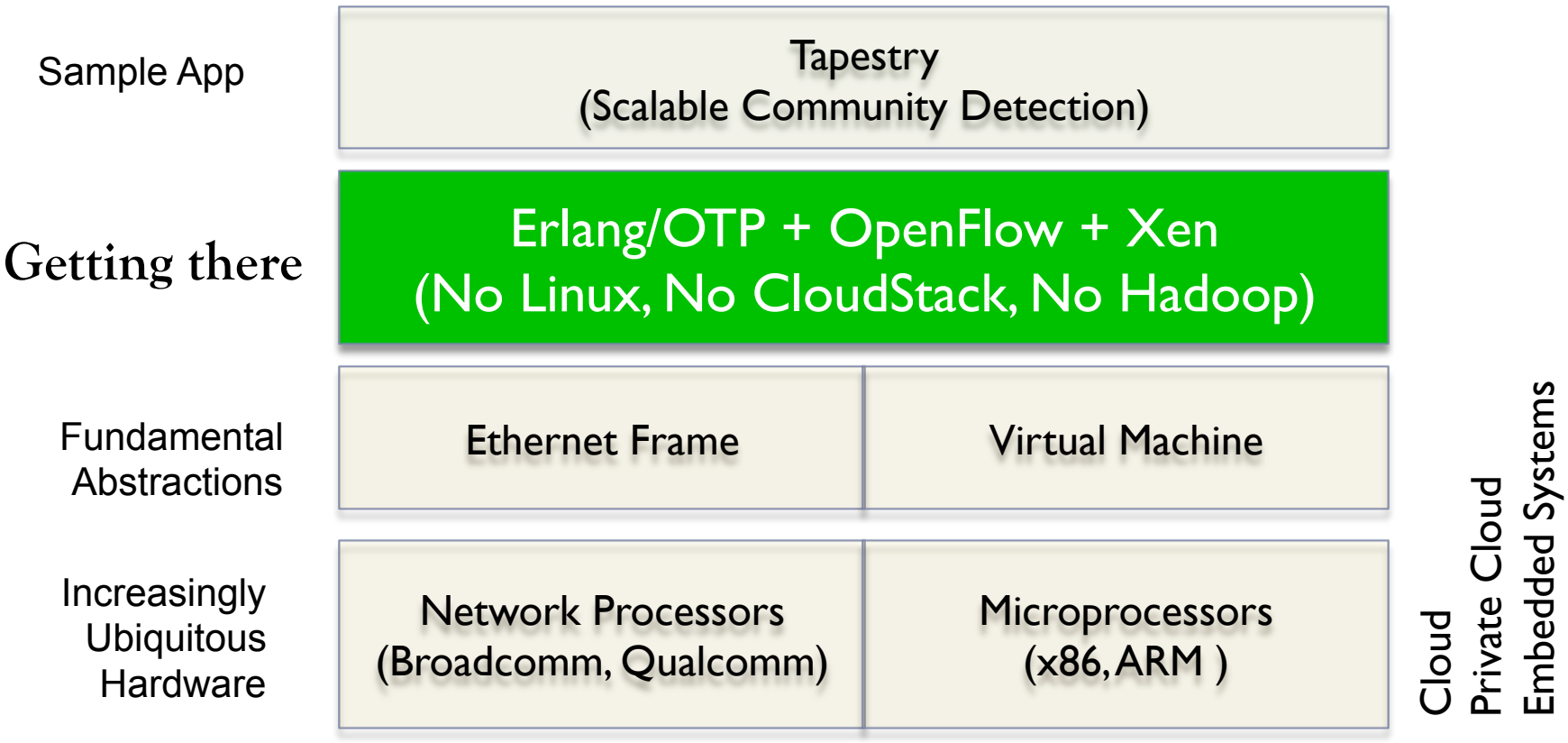
Ethernet + Multi-Core = A New Ubiquitous Machine

"The network is the computer" – John Gage (1984)

We need a better distributed system!



It is not a theory



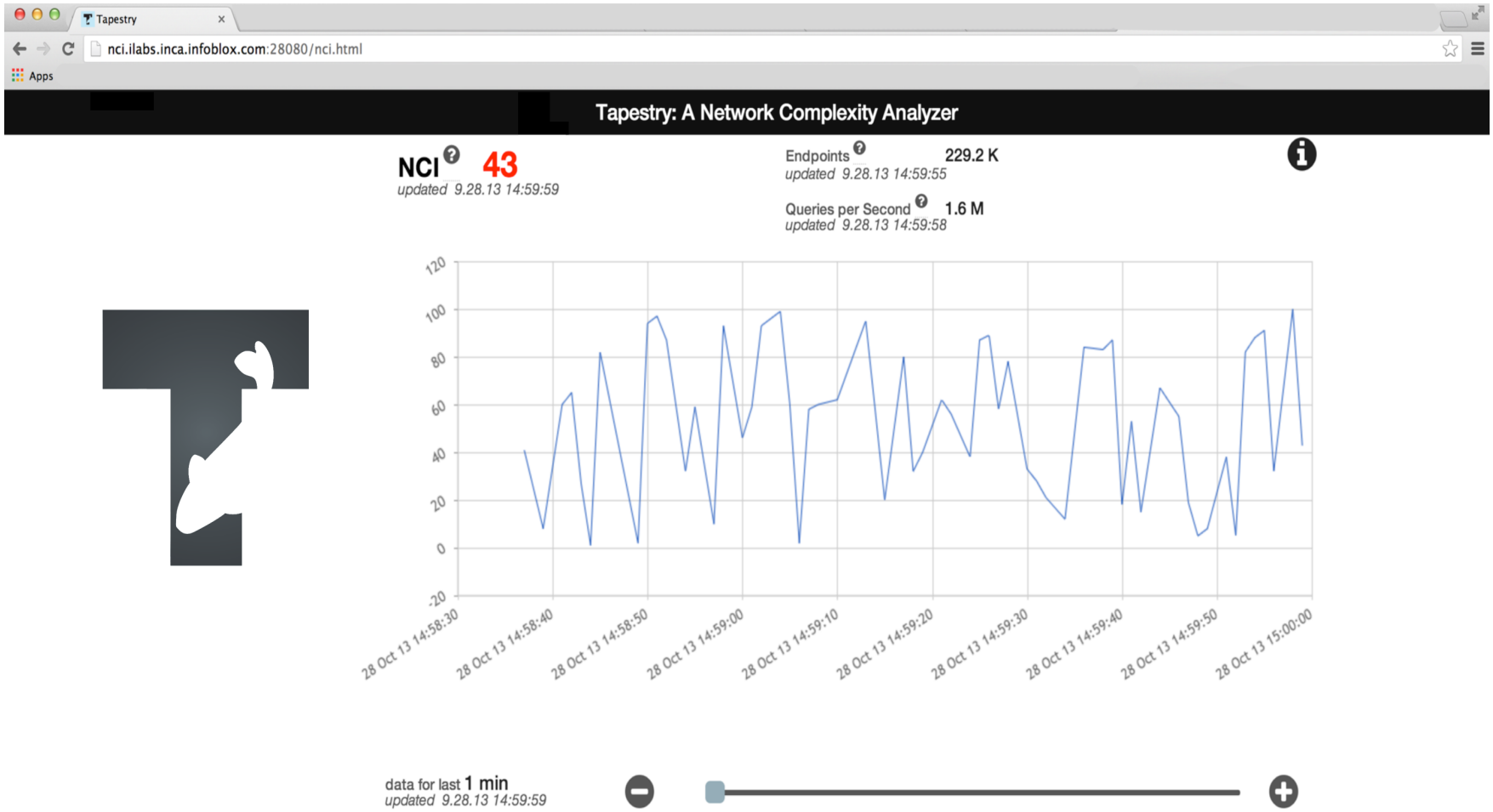
Great but…

Is there
a SINGLE NUMBER that captures

NETWORK COMPLEXITY?

Easy to COMPUTE?

Easy to UNDERSTAND?


Tapestry
nci.ilabs.inca.infoblox.com:28080/nci.html
Apps
Tapestry: A Network Complexity Analyzer
NCI 43
updated 9.28.13 14:59:59
Endpoints 229.2 K
updated 9.28.13 14:59:55
Queries per Second 1.6 M
updated 9.28.13 14:59:58
120
100
80
60
40
20
0
-20
28 Oct 13 14:58:30
28 Oct 13 14:58:40
28 Oct 13 14:58:50
28 Oct 13 14:59:00
28 Oct 13 14:59:10
28 Oct 13 14:59:20
28 Oct 13 14:59:30
28 Oct 13 14:59:40
28 Oct 13 14:59:50
28 Oct 13 15:00:00
data for last 1 min
updated 9.28.13 14:59:59

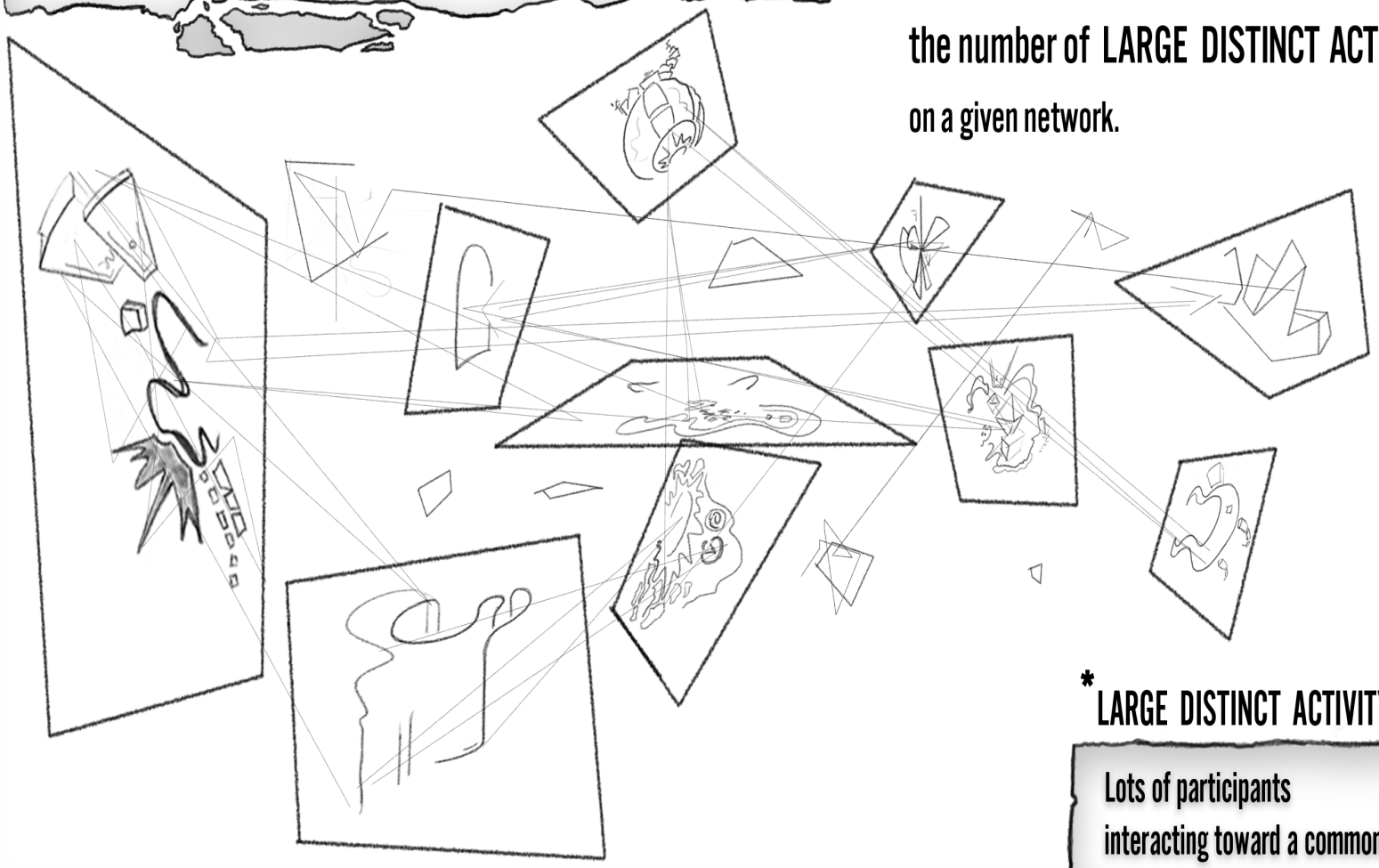
THE NETWORK COMPLEXITY INDEX

A measure

which rises with an increase in

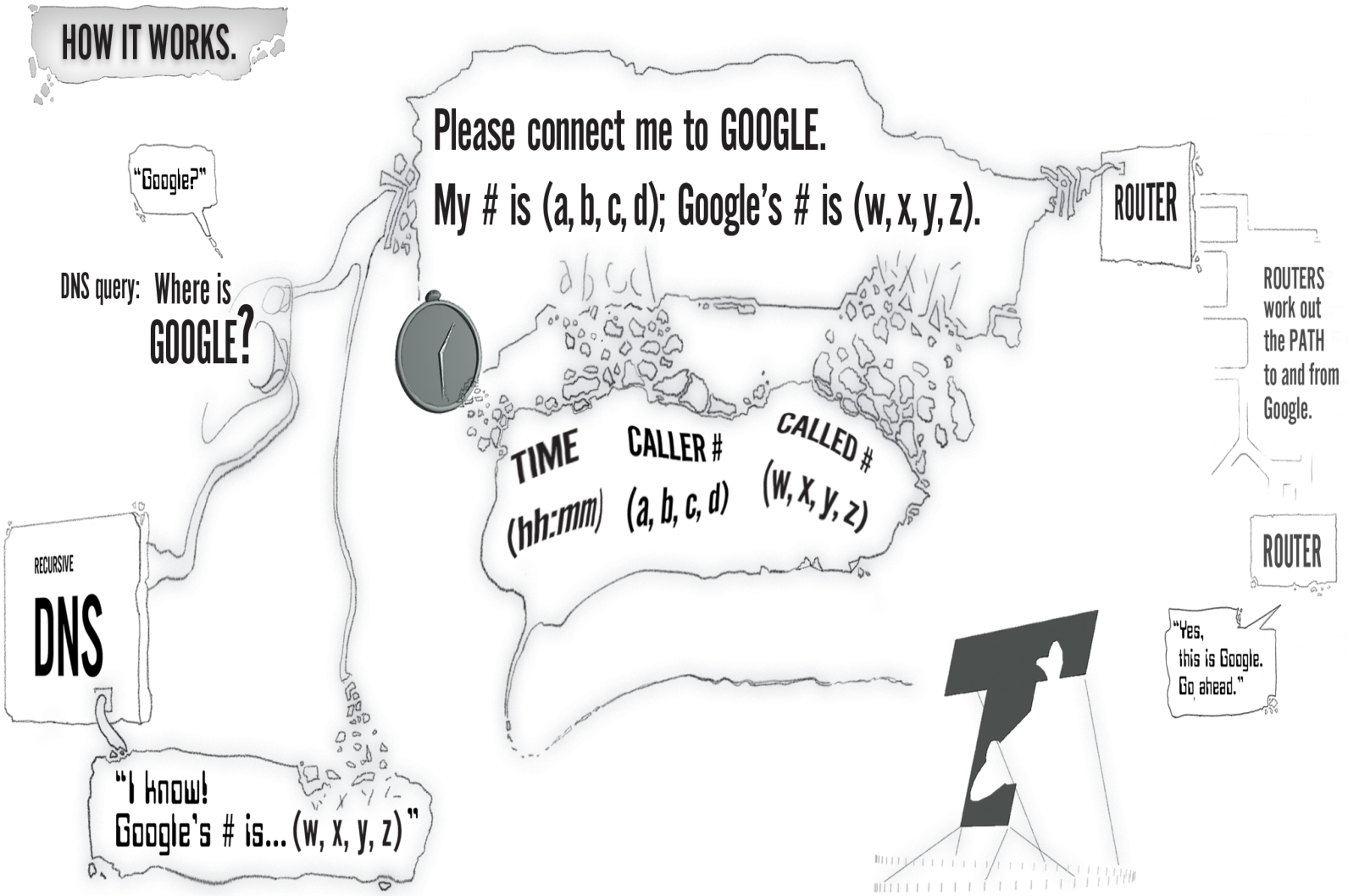
the number of LARGE DISTINCT ACTIVITIES*

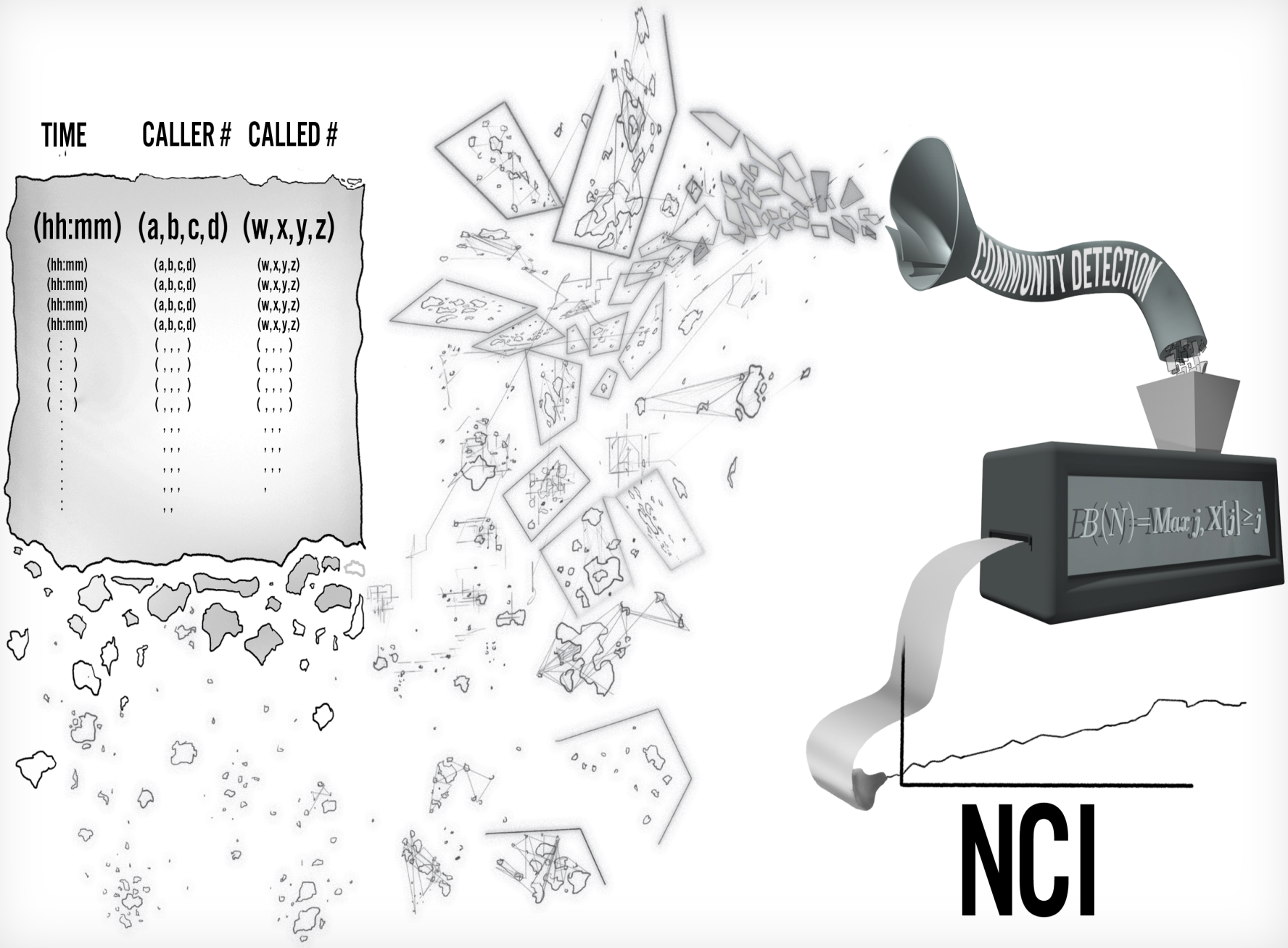
on a given network.



*LARGE DISTINCT ACTIVITY:

Lots of participants
interacting toward a common goal.


HOW IT WORKS.
"Google?"
DNS query: Where is GOOGLE?
RECURSIVE
DNS
"I know! Google's # is... (w, x, y, z)"
Please connect me to GOOGLE.
My # is (a, b, c, d); Google's # is (w, x, y, z).
TIME (hh:mm)
CALLER # (a, b, c, d)
CALLED # (w, x, y, z)
ROUTER
ROUTERS work out the PATH to and from Google.
ROUTER
"Yes, this is Google. Go ahead."

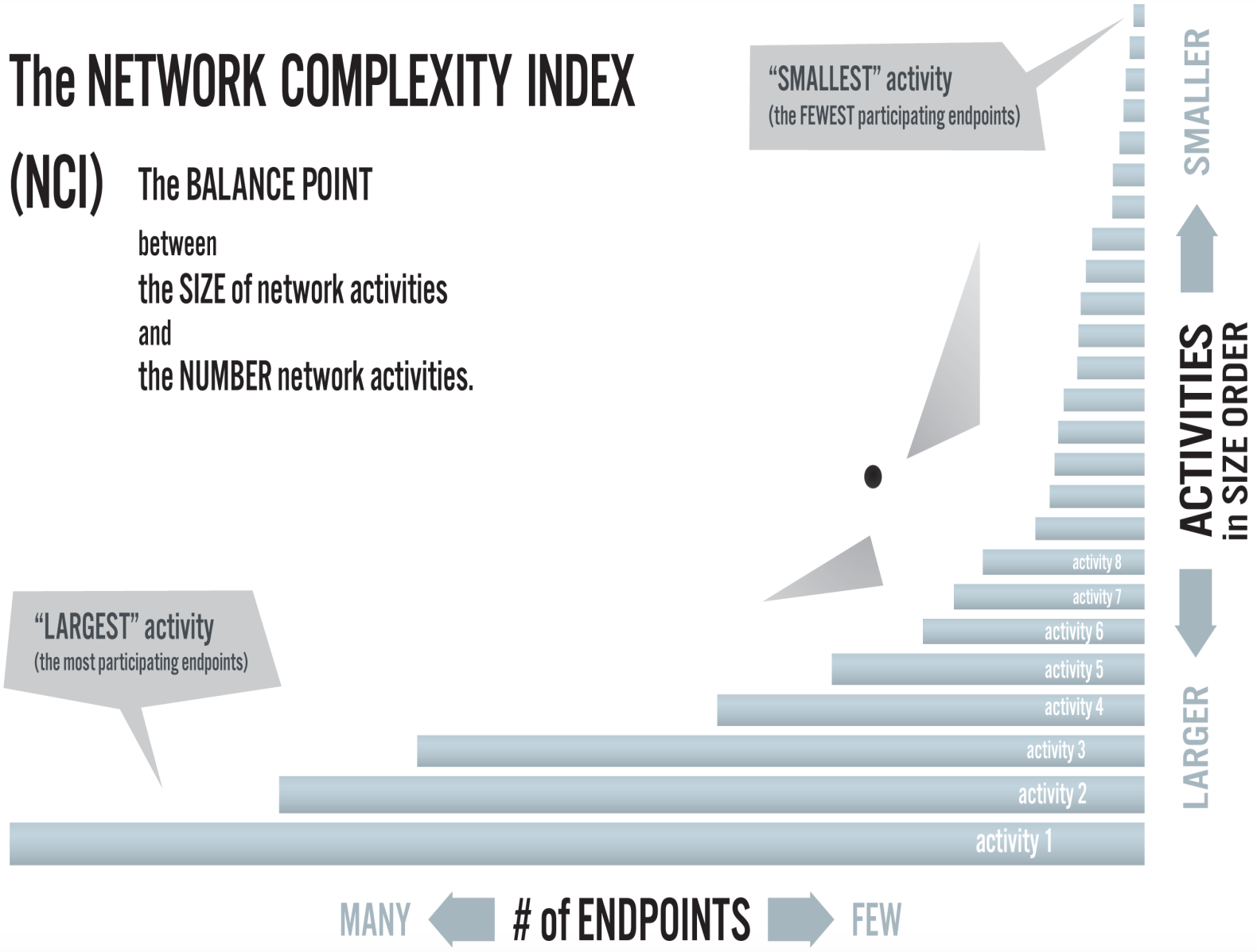

TIME
CALLER #
CALLED #
(hh:mm) (a,b,c,d) (w,x,y,z)
COMMUNITY DETECTION
NCI

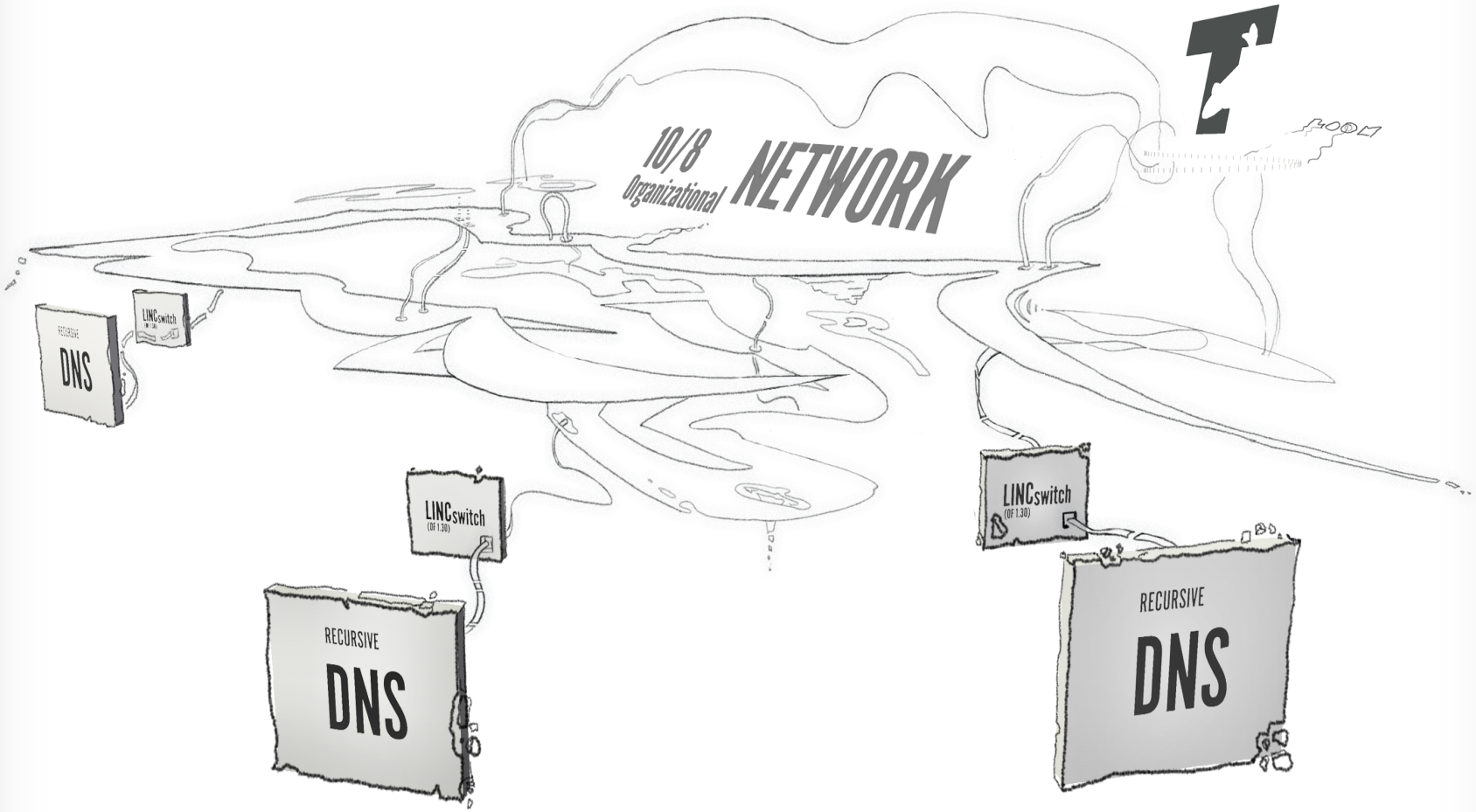
The NETWORK COMPLEXITY INDEX

(NCI)

The BALANCE POINT

between
the SIZE of network activities
and
the NUMBER network activities.




10/8
Organizational
NETWORK
DNS
LINCswitch
(OF1.3.0)
RECURSIVE
DNS
LINCswitch
(OF1.3.0)
RECURSIVE
DNS


CORPORATE
INTRANET
UP-
STREAM ROUTER
10.10.10.2
10.2.45.43-67
10.10.3.49
LINC switch
(OF 1.30)
0 1 2 3
RECURSIVE
DNS
10.10.10.10

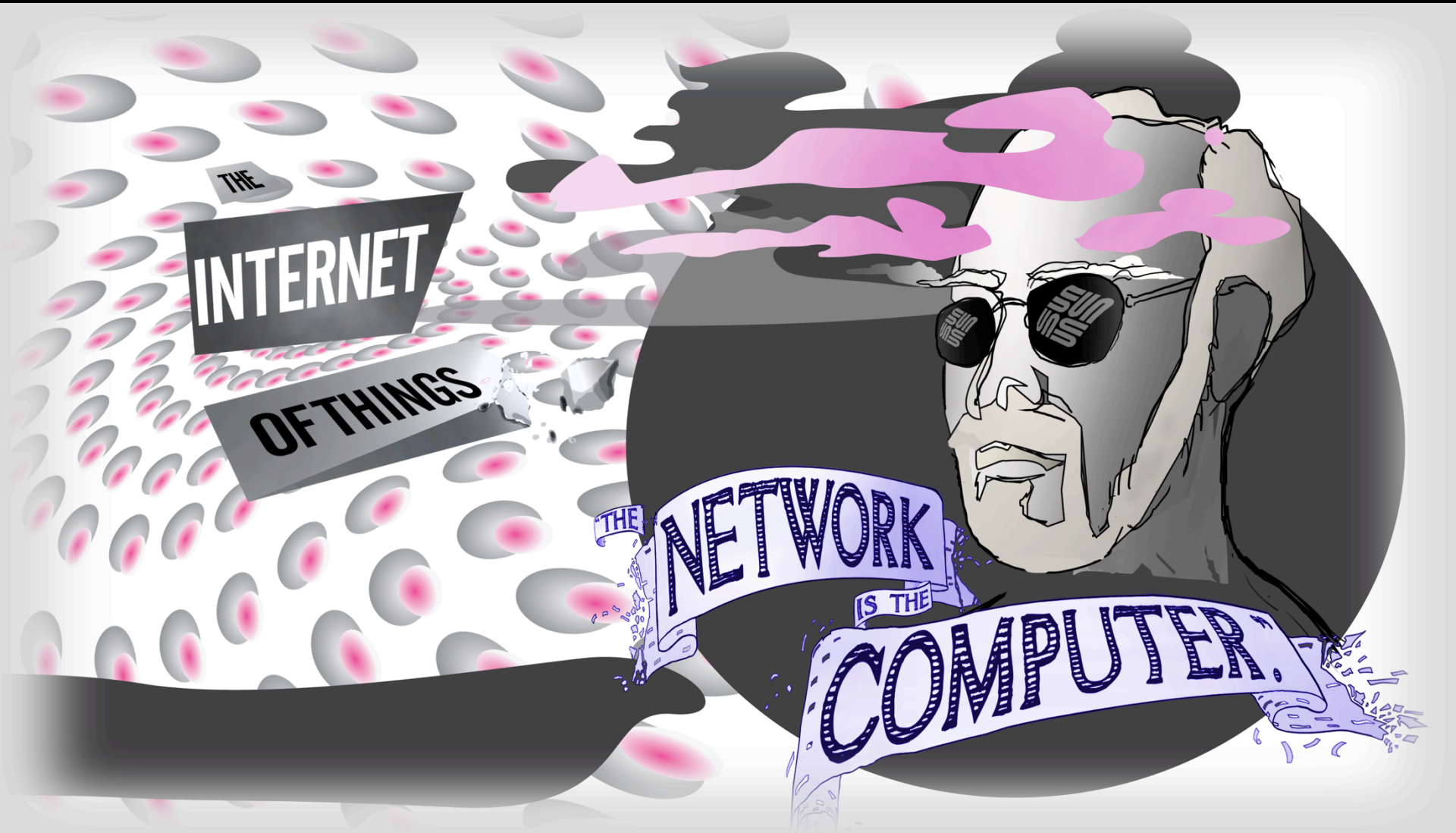
A New Machine?

IF

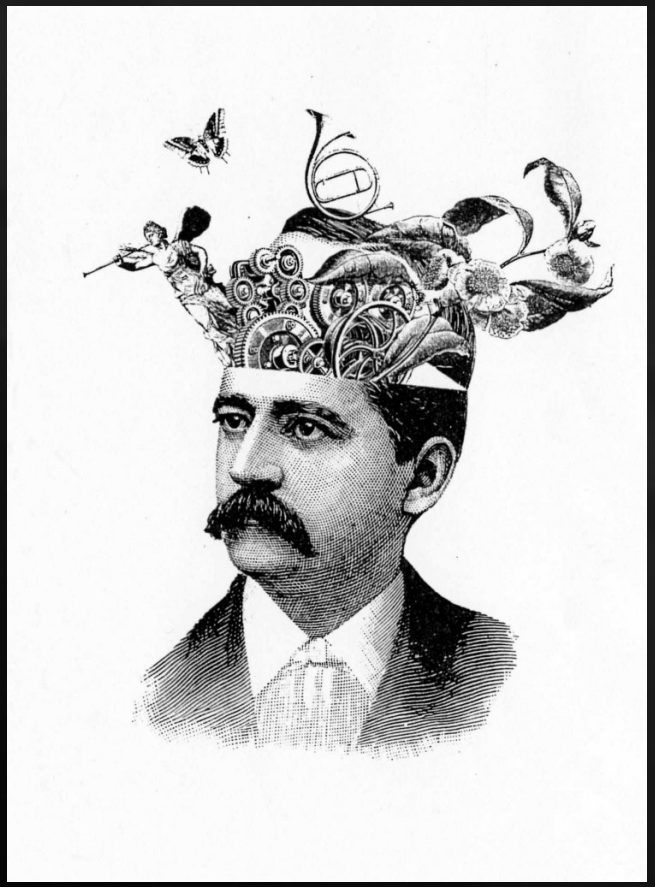
=

Then?

We can look past The Cloud…

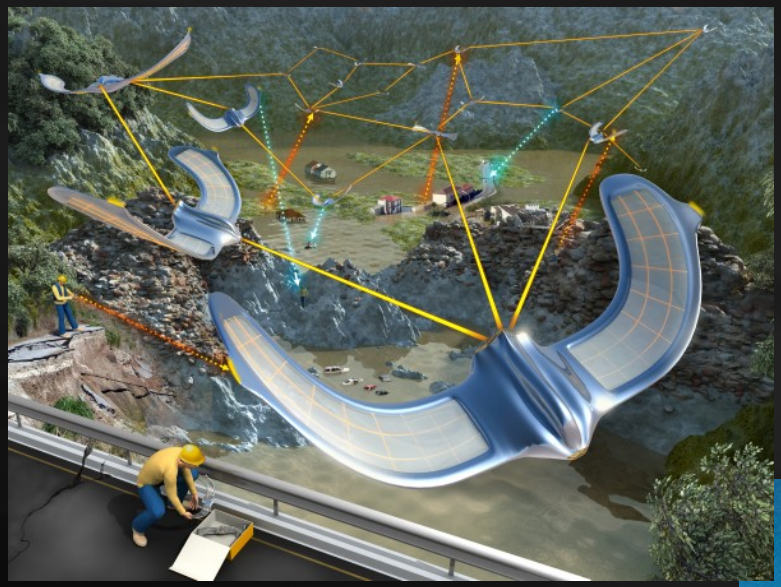


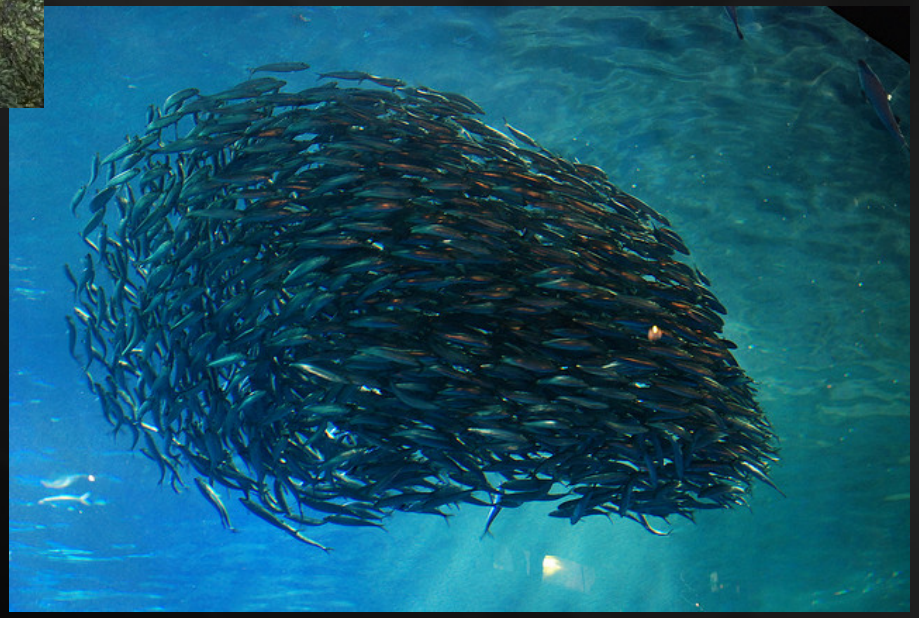
How can an Erlang/OTP developer program the entire network at runtime? (i.e. what abstractions, language extensions, mechanisms, etc.?)



For example, is I/O really a side effect? Is computation king?

Or is communication just as fundamental? Where are Joe's contract checkers??





Help us make it happen!

Email us @ info@FlowForwarding.org

http://www.FlowForwarding.org