The Internet without IP Addresses

Bruce Simpson
School of Computer Science
University of St Andrews
http://www.cs.st-andrews.ac.uk/~bms/

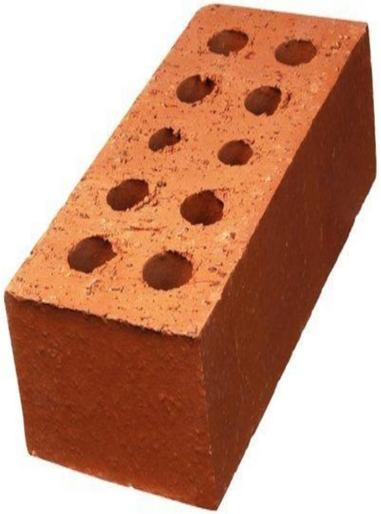
PhD Year 1
Supervised by Saleem Bhatti
Sponsored by Cisco Systems, Inc.

What this talk is about

- Problem background
- Eliminating IP Addresses
  - Systems-level challenges
- Can we build it?

Internet (With Fries)

- We wish to try and support a ***harmonised solution to many* network functions**:
  - Host & Site Multi-homing
  - Host & Network Mobility
  - Multi-path capable transports: MP-TCP
  - Localised addressing: NAT
  - Traffic Engineering
  - Packet-level, end-to-end security.



- Current solutions use separate components
  – They lack *functional integration*

Separate Semantics

- Semantic overload of IP address:
  - **locator** semantics + **identifier** semantics
  - Ease implementation of mobility, multi-homing…
- New architecture, old problem:
  - RFC4984 (IAB, 2007) and RFC2101 (IAB, 1997)
  - IEN1 (1977), IEN19 (1978), IEN23 (1978)
- Many solutions now proposed:
  - HIP, LISP, SHIM, SixOne
  - **Identifier-Locator Networking Protocol (ILNP)**

Naming: IP vs. ILNP

Protocol Layer	IP	ILNP
Application	FQDN or IP address	FQDN (RFC1958)
Transport	IP address (+ port number)	Identifier (+ port number)
Network	IP address	Locator
(Interface)	IP address	(dynamic binding)

Entanglement **Separation**

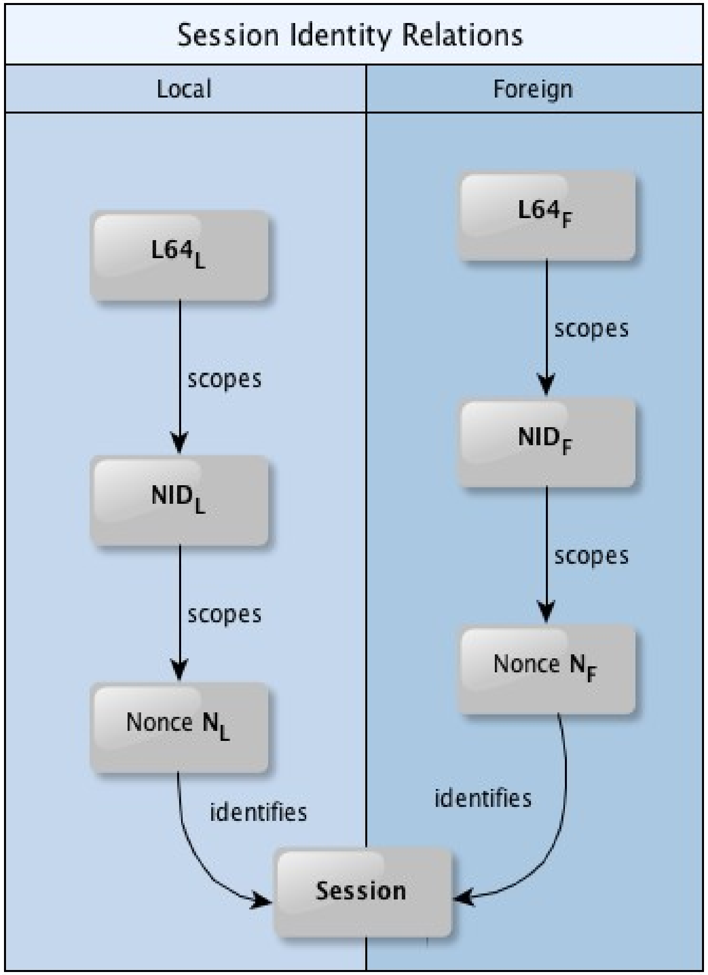
FQDN = fully qualified domain name

Systems Challenges

- Introduce new namespace into OS stack:
  - As a superset of IPv6 – **ILNPv6**
  - Separate semantics of naming functions
- Dis-entangle stack layers:
  - Implicit bindings due to use of IPv6 address
- Make bindings 1:N and dynamic:
  - Between transport and network layer
  - Between network layer and interface
- Protocol level:
  - Encode I and L values into current packets

ILNPv6 Locators & Identifiers

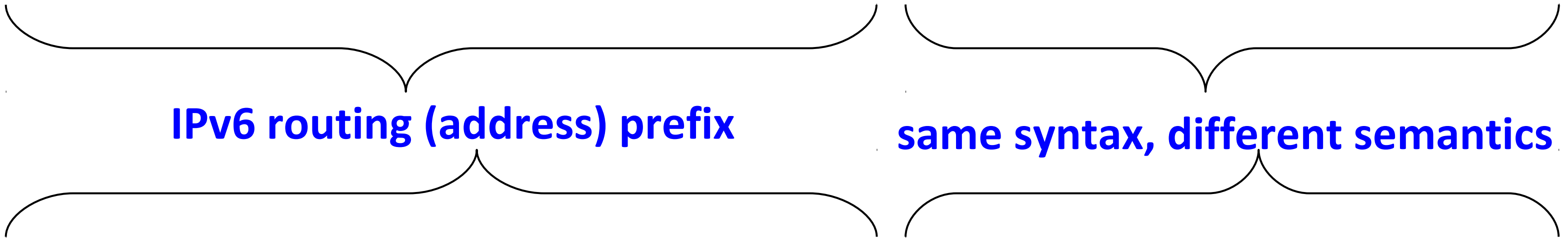
- ILNP functions manage:
  - Locator (L64) values
  - Node IDs (NIDs)
  - <L64, NID> bindings
- ILNP end-to-end state:
  - bound through NID values
  - <L64, NID> relation is 1:N
- Bindings are now **dynamic**



IPv6 addresses & ILNPv6

```
IPv6 (as in RFC3587 with RFC3177 - RFC6177 has 8-bit subnet ID):

| 3 |     45 bits         |  16 bits  |      64 bits              |
+---+---------------------+-----------+---------------------------+
|001|global routing prefix| subnet ID |    Interface Identifier   |
+---+---------------------+-----------+---------------------------+
```



IPv6 routing (address) prefix

same syntax, different semantics

```
ILNPv6:

|             64 bits                 |      64 bits              |
+---+---------------------+-----------+---------------------------+
|             Locator                 |      Node Identifier      |
+---+---------------------+-----------+---------------------------+
```

same syntax and semantics as
IPv6 routing (address) prefix
so IPv6 core routers work as today

these bits only examined and
acted upon by end systems

IPv6 Datagrams

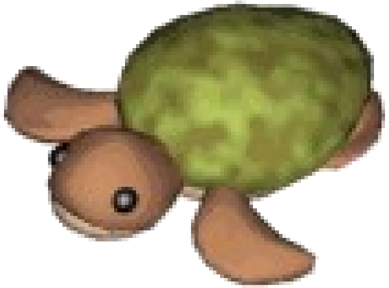
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |   Next Hdr    |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+-                       Source Address                        -+
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+-                    Destination Address                      -+
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ILNPv6 Datagrams

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |   Next Hdr    |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                        Source Locator                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                      Source Identifier                        +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                      Destination Locator                      +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                    Destination Identifier                     +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

How do we build it?

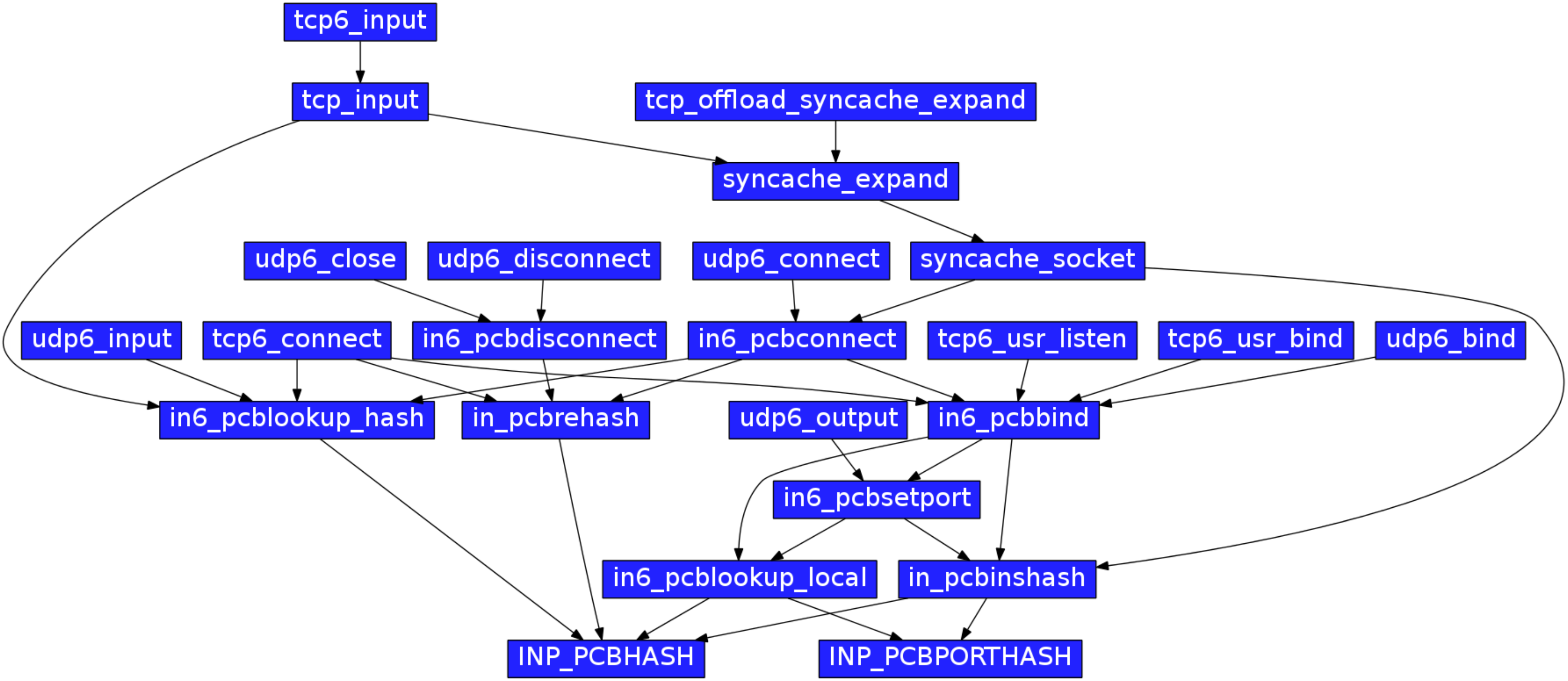
- "Easy – just update KAME, watch the turtle dance some more ..." (http://www.kame.net/) BZZT- Wrong.
- However, many issues with FreeBSD netinet6:
  - kernel state bindings
  - code/data paths for IPv6 packets
  - socket bindings and socket state



- ...or indeed any network stack:
  - Assumptions re IPv6 address usage, *interface identifiers*
  - Socket *address space*
- *...Systems-level challenge (complexity, depth)*

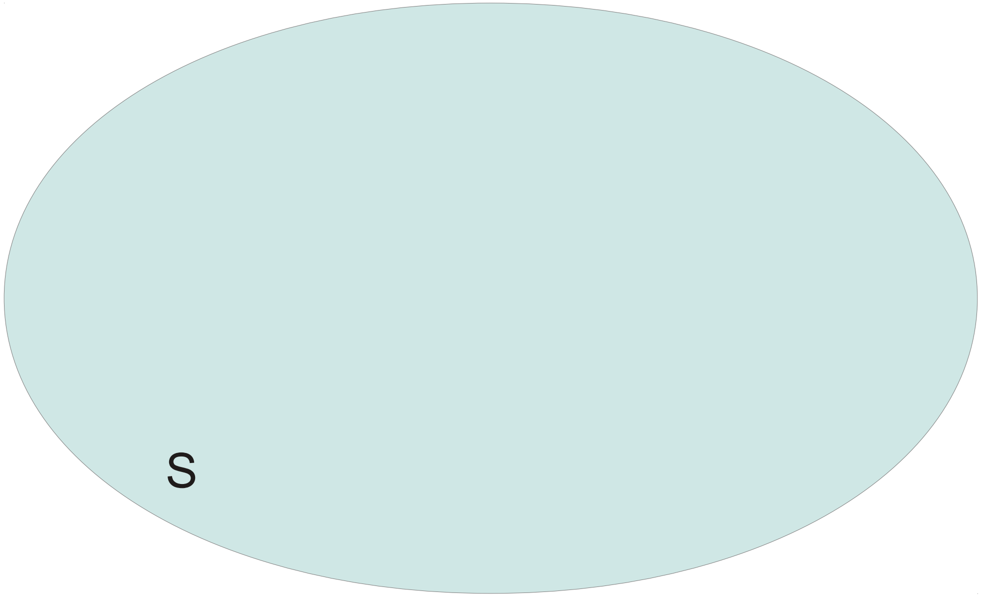
The binding problem

Where do we put ILNP?



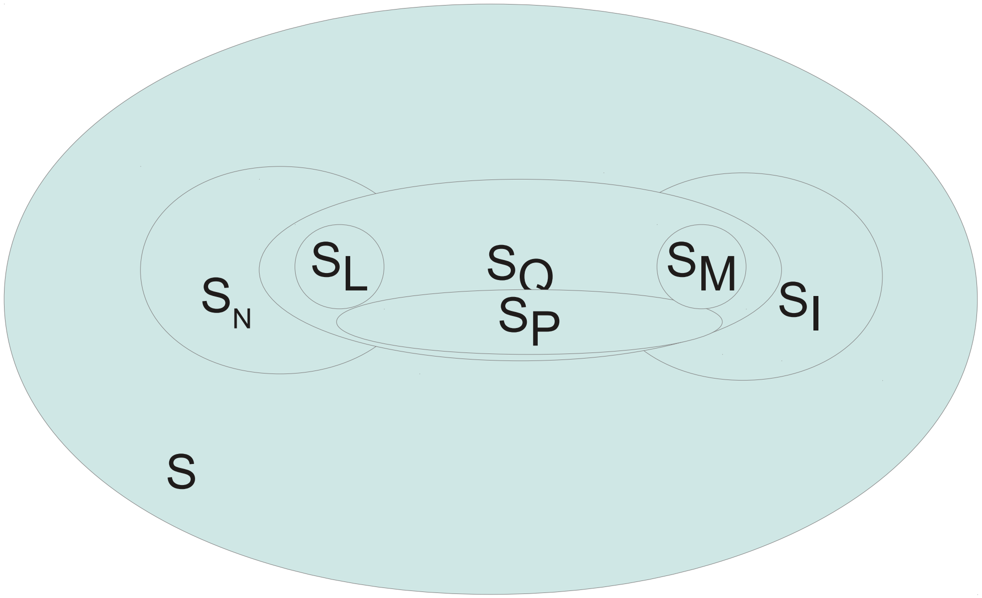
Sockets in IPv6

S	All sockets



Sockets in dual-stack IPv6/ILNPv6

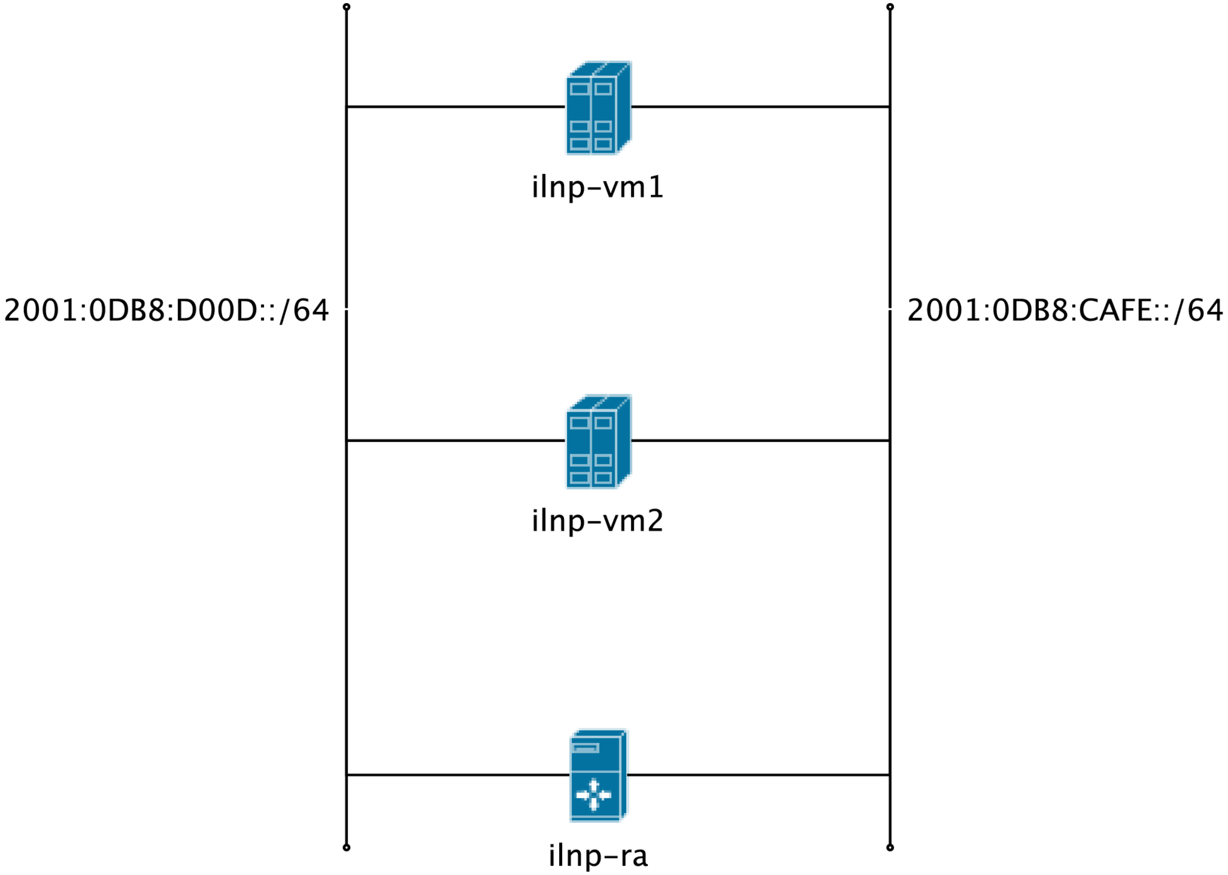
S	All sockets
S_I	ILNP sockets
S_N	Non-ILNP sockets
S_P	Sockets bound to local port NID
S_Q	Sockets bound to local port NID, wildcard
S_L	Non-ILNP sockets bound to local port NID
S_M	ILNP sockets bound to a NID



The research issue is **binding dynamicity**

Yes we can!

Demonstration



Demonstration

- Questions:
  - – Will legacy apps interoperate with ILNPv6?
  - – Stack is modified – do we change apps too?
- Answer to both questions:
  - – Yes, and no ;-)
- Demo – unmodified *ping6* binary
  - – Transparent ILNPv6 operation below raw socket
    - Locator Updates handled by kernel
    - Provides failover, mobile handover, multi-homing…

Thank you

- Questions? Comments?
  - Contact bms@ for demo
- Source code available Dec2012/Jan2013
- Project web site

  `http://ilnp.cs.st-andrews.ac.uk/`

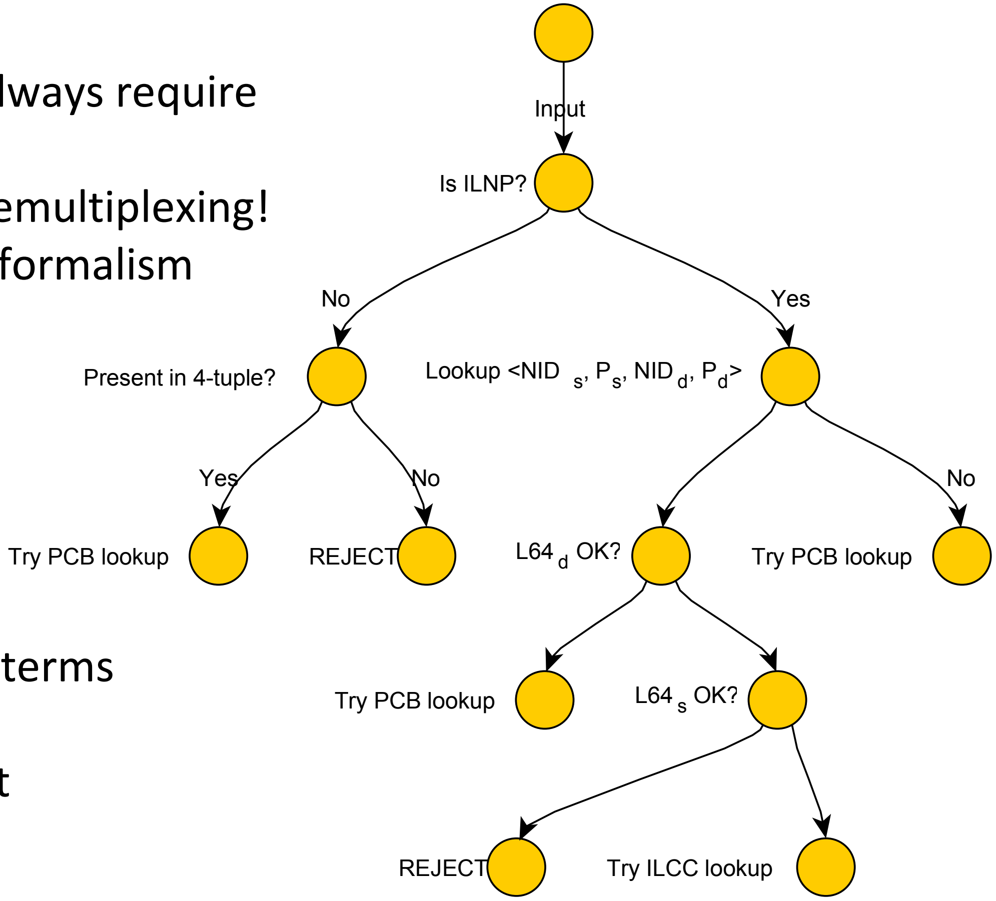
Bruce Simpson
School of Computer Science
University of St Andrews
http://www.cs.st-andrews.ac.uk/~bms/

Input Classification

- TCP, UDP in ILNPv6 should not always require ILNPv6 headers
  - Poses problems for input demultiplexing!
  - Does not lend itself well to formalism (at this stage)

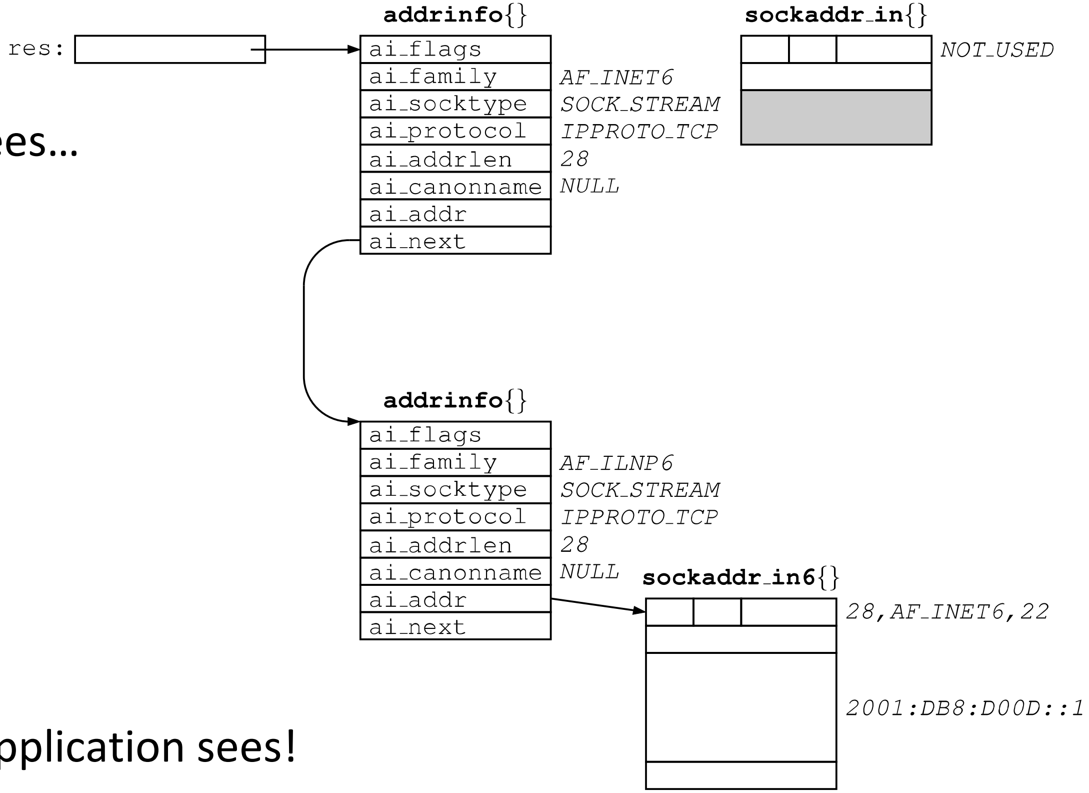
- Use hash tables

- Problem framed in IPv6/ILNPv6 terms
  - Ease development now
  - Inform future development



Resolver changes

Change what the resolver sees…



res:

addrinfo{}

ai_flags	
ai_family	*AF_INET6*
ai_socktype	*SOCK_STREAM*
ai_protocol	*IPPROTO_TCP*
ai_addrlen	*28*
ai_canonname	*NULL*
ai_addr	
ai_next	

sockaddr_in{}

NOT_USED

addrinfo{}

ai_flags	
ai_family	*AF_ILNP6*
ai_socktype	*SOCK_STREAM*
ai_protocol	*IPPROTO_TCP*
ai_addrlen	*28*
ai_canonname	*NULL*
ai_addr	
ai_next	

sockaddr_in6{}

28,AF_INET6,22

2001:DB8:D00D::1

…and you change what the application sees!

Figure 1: Example of information returned by an ILNP-aware **getaddrinfo**.

Future work

- Not everything has a "name"
- P2P
  - – Bitcoin, Bittorrent, RTMP
    - Special case of naming
- IP Multicast
  - – Each *group* scoped to *rendezvous point*