SHISA: The Mobile IPv6/NEMO BS Stack Implementation Current Status

Asia BSD Conference 2007
11th March 2007 @ Tokyo, Japan

Keiichi Shima[1], Koshiro Mitsuya[2], Ryuji Wakikawa[2], Tsuyoshi Momose[3] and Keisuke Uehara[2]

[1]Internet Initiative Japan Inc. [2]Keio University [3]NEC Corporation

Topics

- Mobile IPv6/NEMO BS Basics
- SHISA History
- SHISA Design
- Implementation
- Consideration
- Future Plans

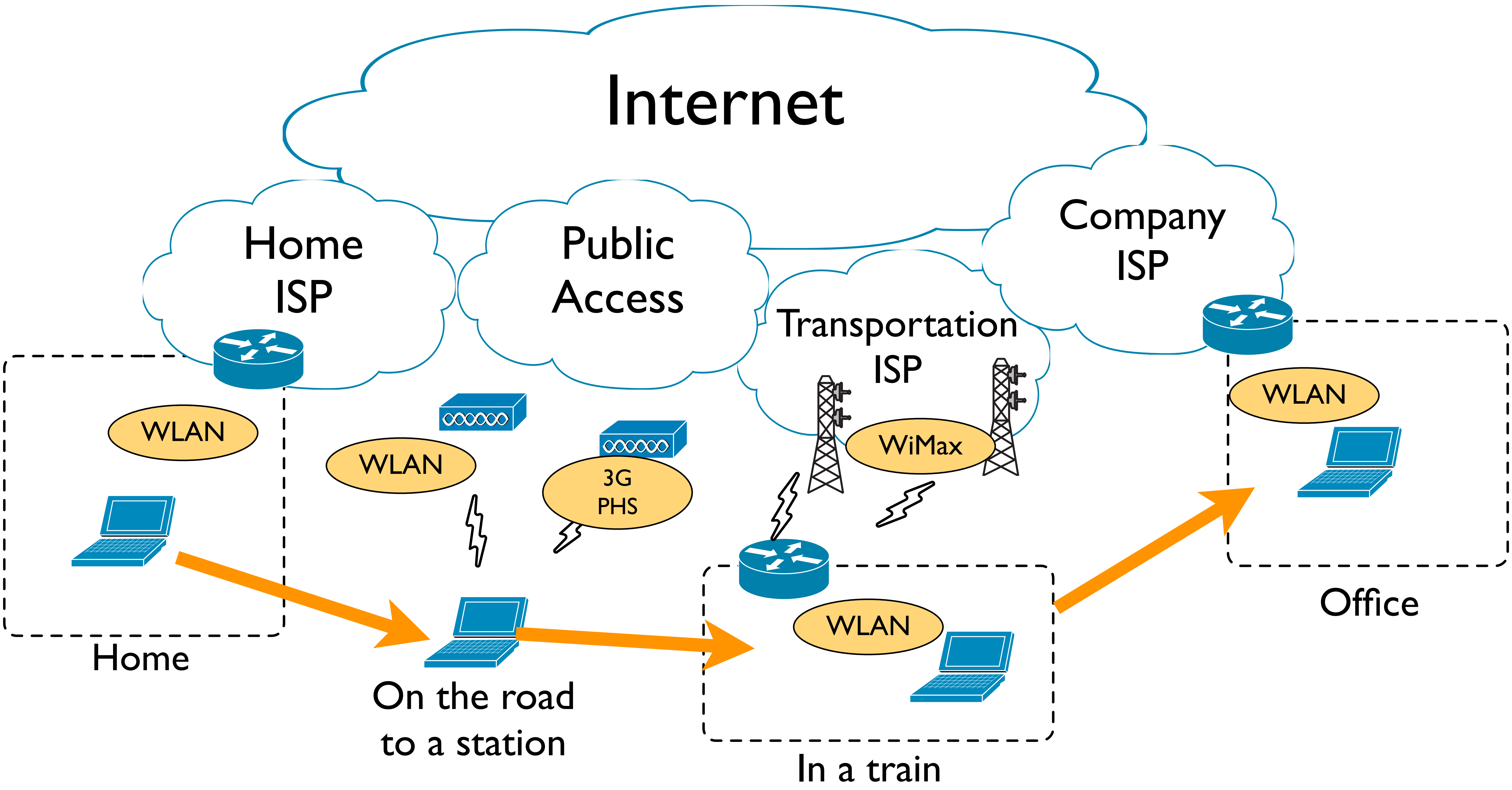
Mobile IPv6/NEMO BS

- Movement (address change) is hidden in the IPv6 layer
  - A node can move between different communication media
  - No modification to the transport layer and above

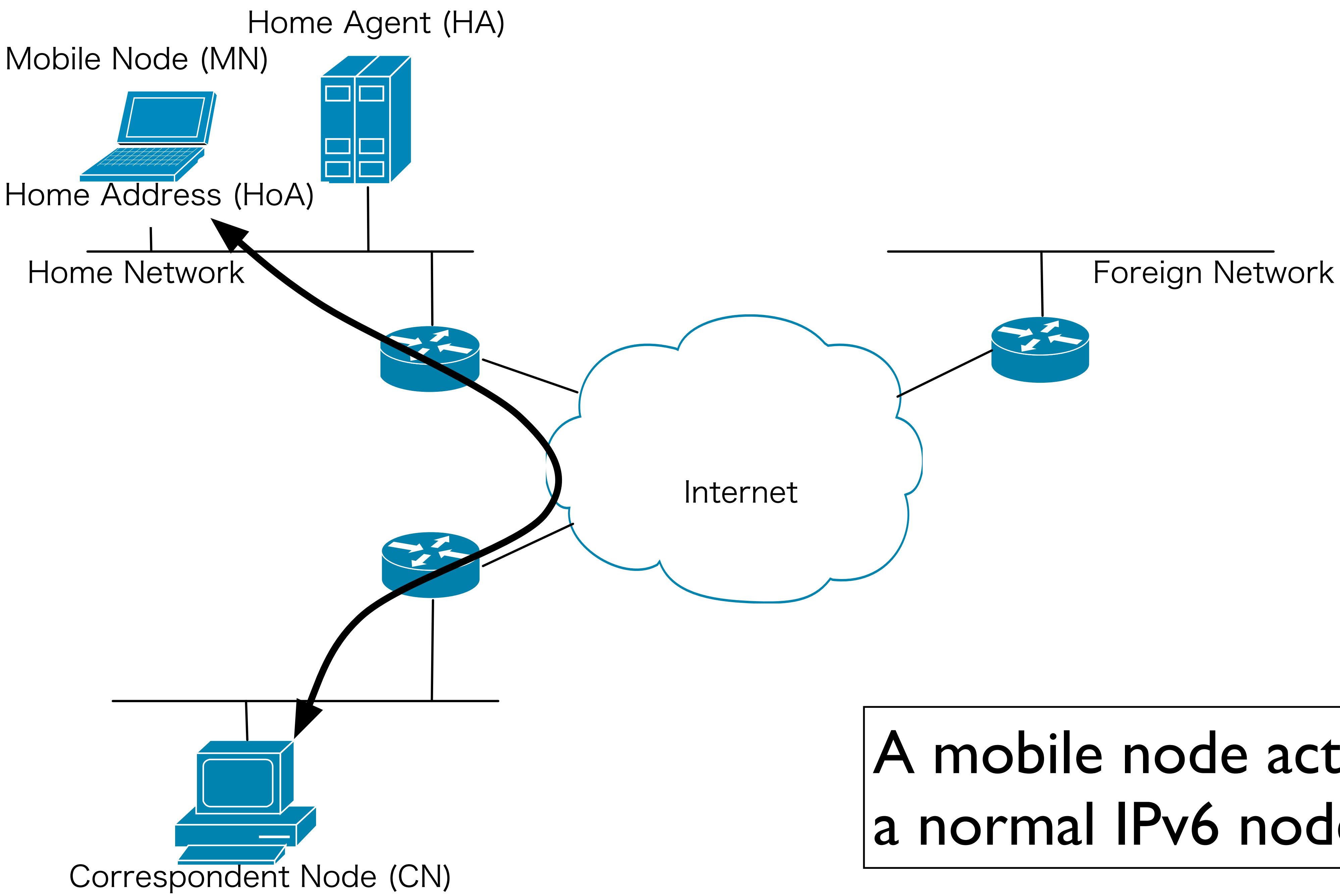
Why Important?

- Wireless broadband Internet
- Built-in communication devices
- Always connected environment
- Application areas
  - Next generation mobile phones
  - Transportation (trains, buses, aviation)
  - Personal Mobile Router

Future un-wired Internet

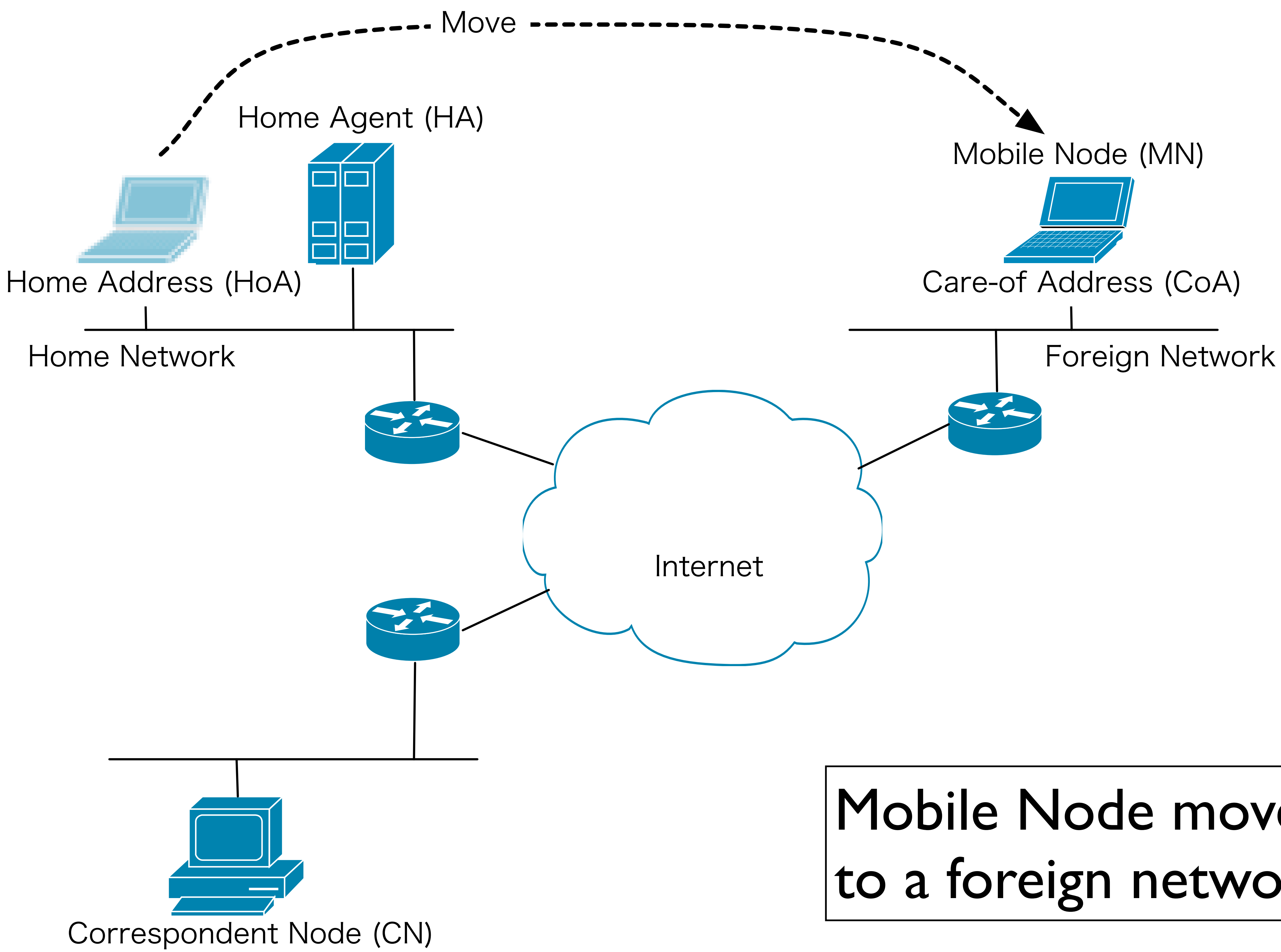


Mobile IPv6 Overview



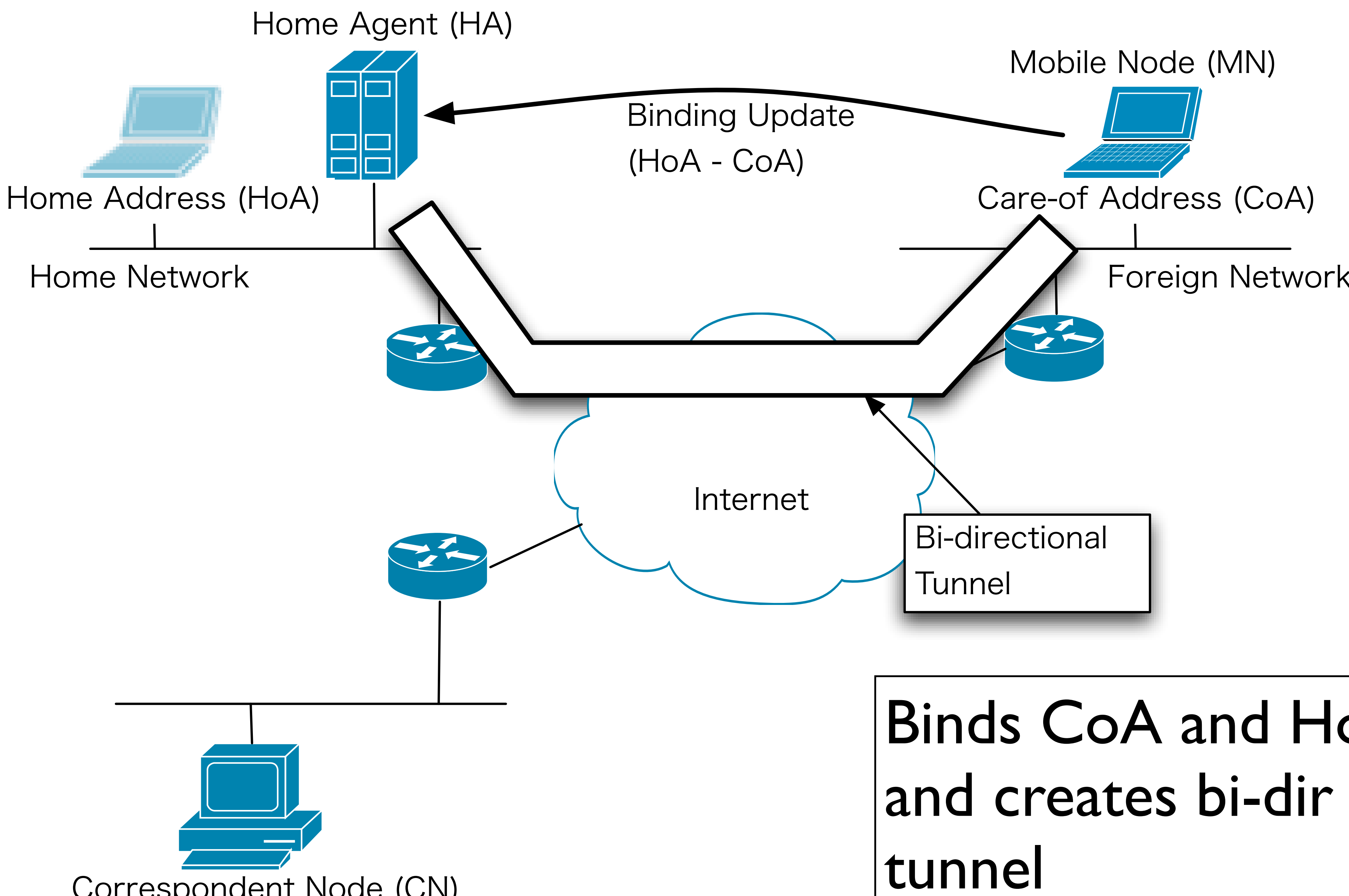
A mobile node acts as a normal IPv6 node

Mobile IPv6 Overview



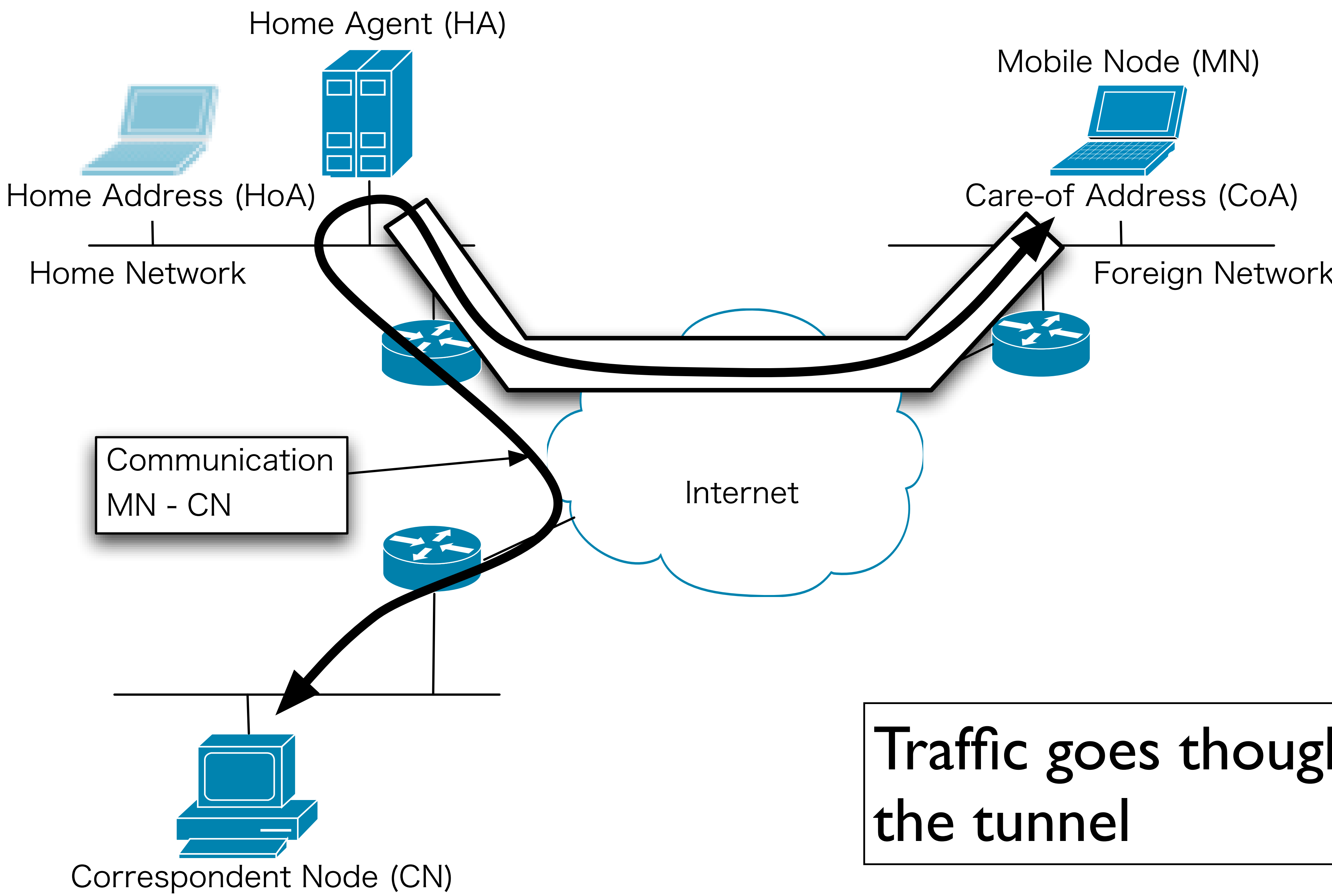
Mobile Node moves to a foreign network

Mobile IPv6 Overview



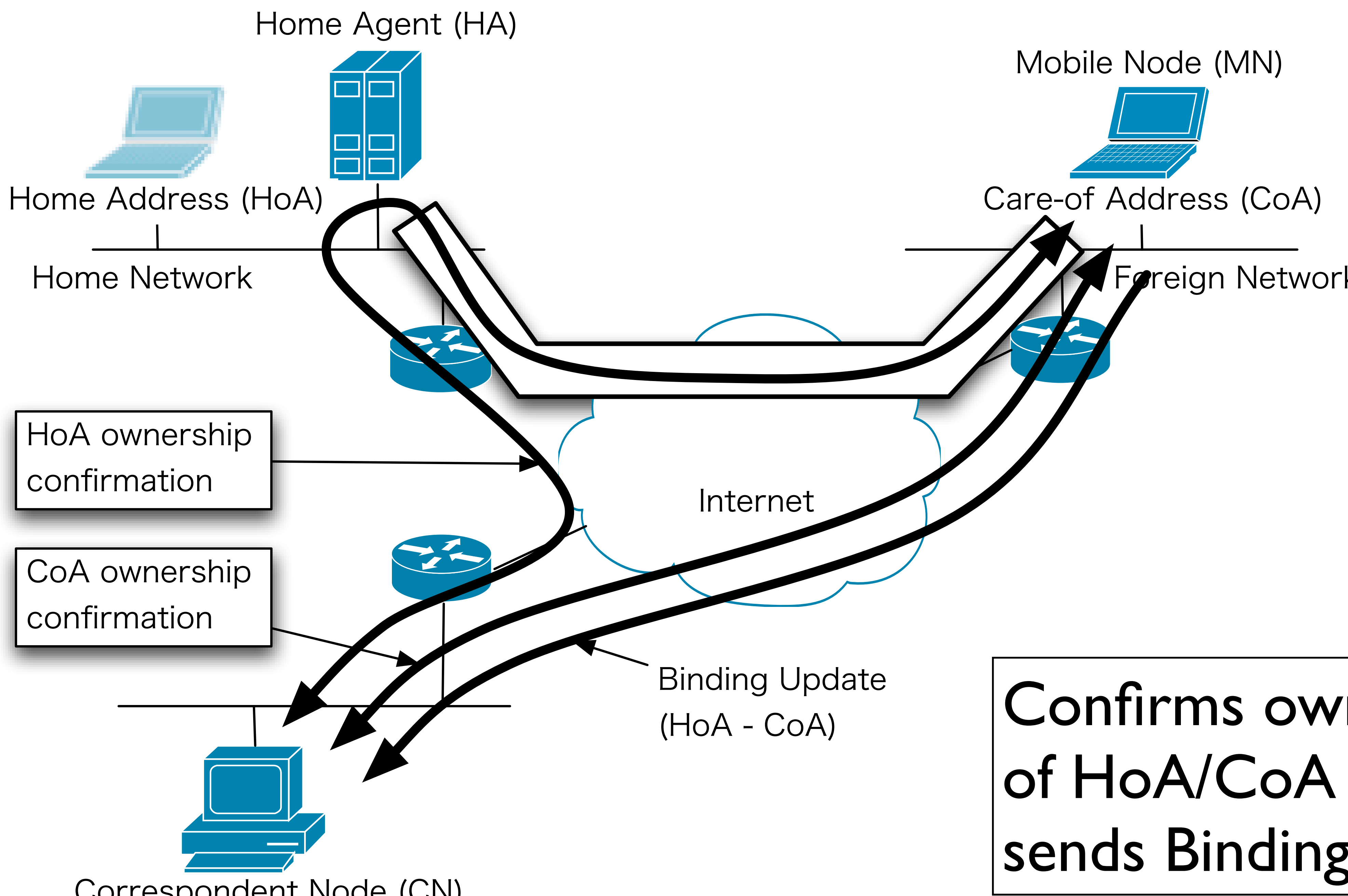
Binds CoA and HoA and creates bi-dir tunnel

Mobile IPv6 Overview



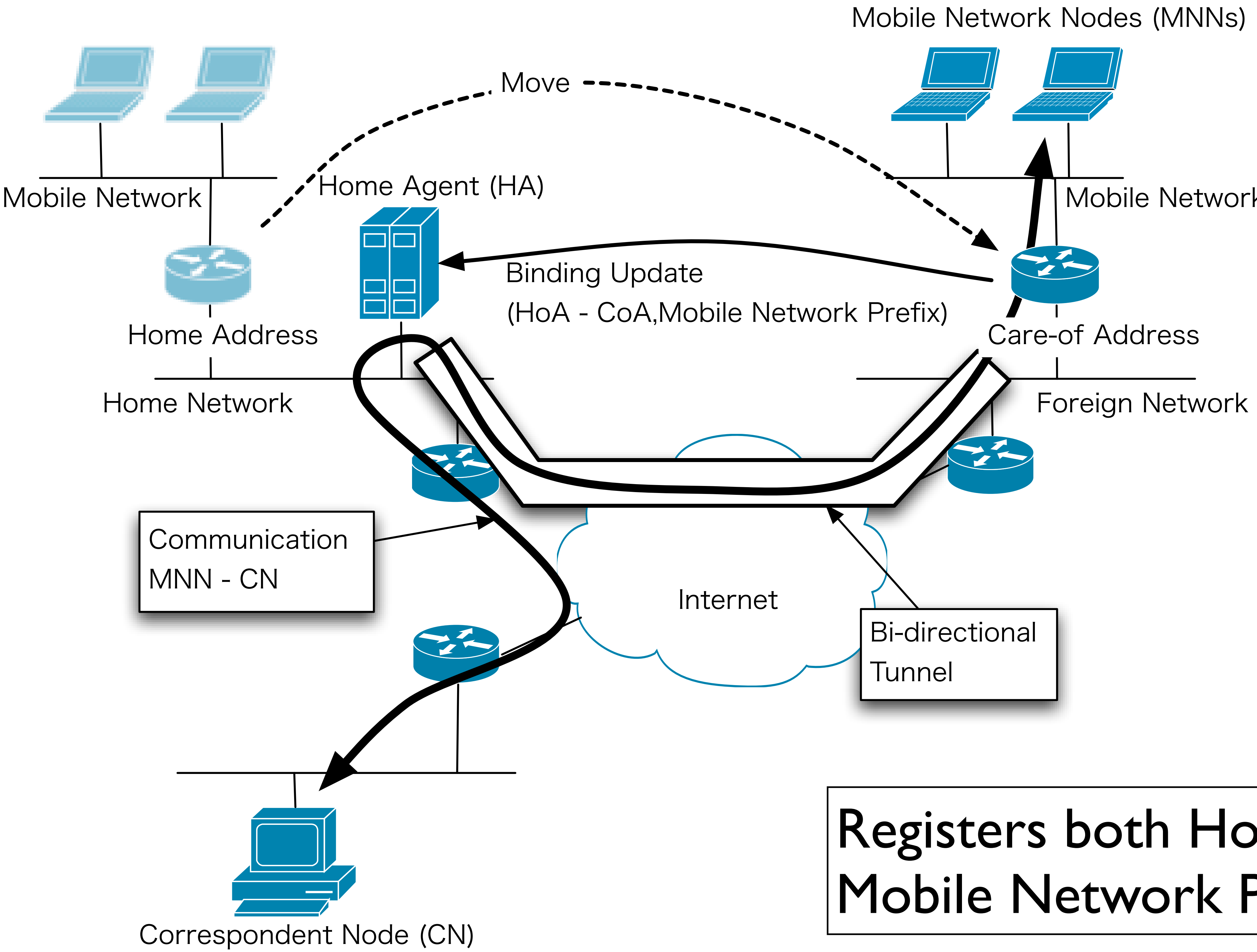
Traffic goes though the tunnel

Mobile IPv6 Overview



Confirms ownership of HoA/CoA and sends Binding Update

NEMO BS Overview



Registers both HoA and Mobile Network Prefix

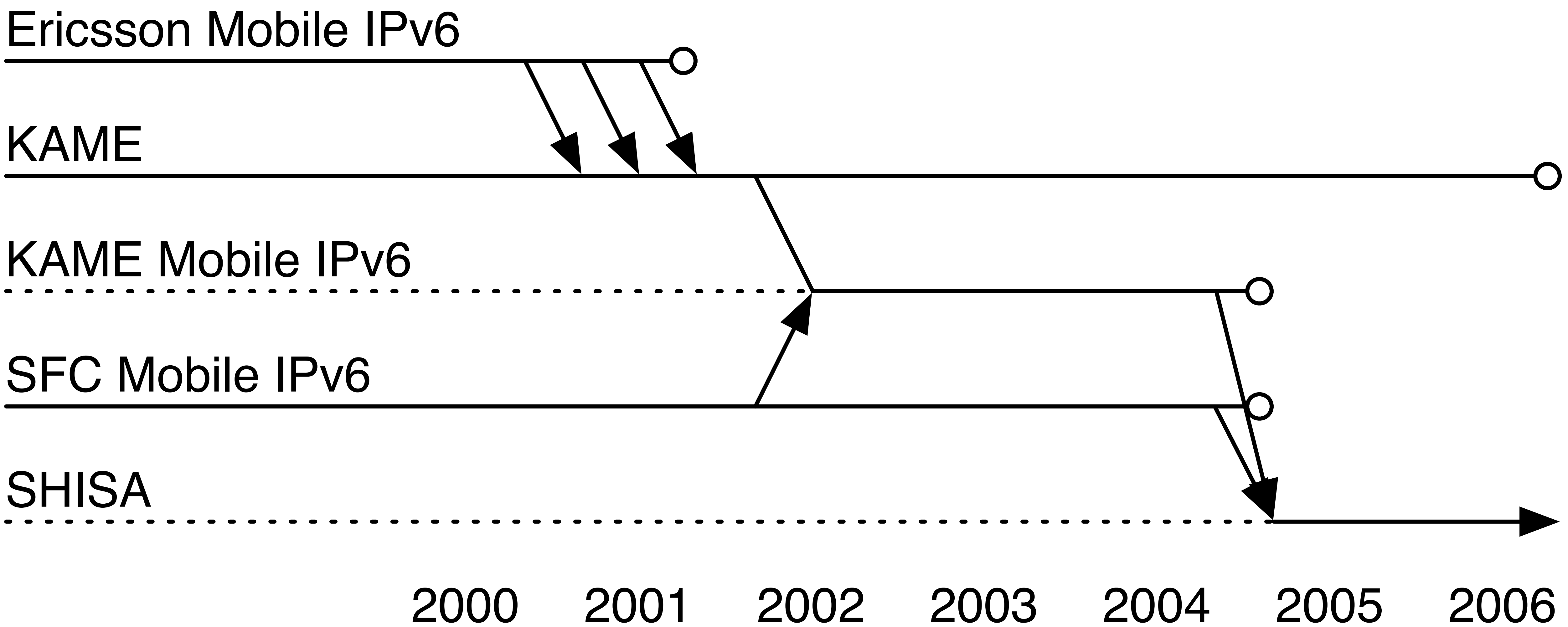
Objectives

- Deploy IPv6 mobility
  - A free working code as a reference code is important for deployment
    - ex1) the TCP/IP code by UCB
    - ex2) the KAME IPv6 code
- This presentation introduces our implementation, its design and current status

SHISA

- A free Mobile IPv6 / NEMO BS stack for BSD operating systems
- The project started as a part of the KAME project and launched as a separate project after the KAME project concluded
- NetBSD 2.0 and FreeBSD 5.4R (and OpenBSD 3.0 partially) were originally supported
  - We started porting works to the original BSDs
  - NetBSD-current is our first target

SHISA History



SHISA Features

- Mobile IPv6 (RFC3775)
  - Mobile Node, Home Agent, Correspondent Node
  - Including Route Optimization
- NEMO BS (RFC3963)
- Multiple Care-of Addresses Registration (based on the older draft)
- Dual Stack Mobile IPv6 (based on the -01 draft)

SHISA Design

- Easier development
- Adaptability to various network movement detection mechanisms
- Simple application interface
- Minimum modification of kernel code

Easier Development

- We wanted to move the code to user space
  - Destination Opt v.s. Mobility Header
- Separate signal processing part and packet forwarding processing part
  - Signal processing is done in user space programs
  - Packet forwarding is done in the kernel
- Similar to the BSD routing mechanism

Easier Development

- Signal processing is too much to implement in the kernel
- We can use various debugging tools for user space programs
- Bigger number of user space application developers than kernel developers

Adaptability

- The requirements of mobile device movement detection may vary based on the technologies of mobile carriers
- Movement detection mechanism is implemented as a separate program so that each operator can replace the program

Easier Application Interface

- Mobility activities can be monitored by the special socket interface
- All mobility kernel function can be controlled with the socket interface
- Similar to the Routing Socket

Minimum Modification of the Kernel

- The final goal of our project is to merge the mobility function to the original BSD operating systems
- The modification of the kernel should be minimized to make the integration work easier

Program Organization

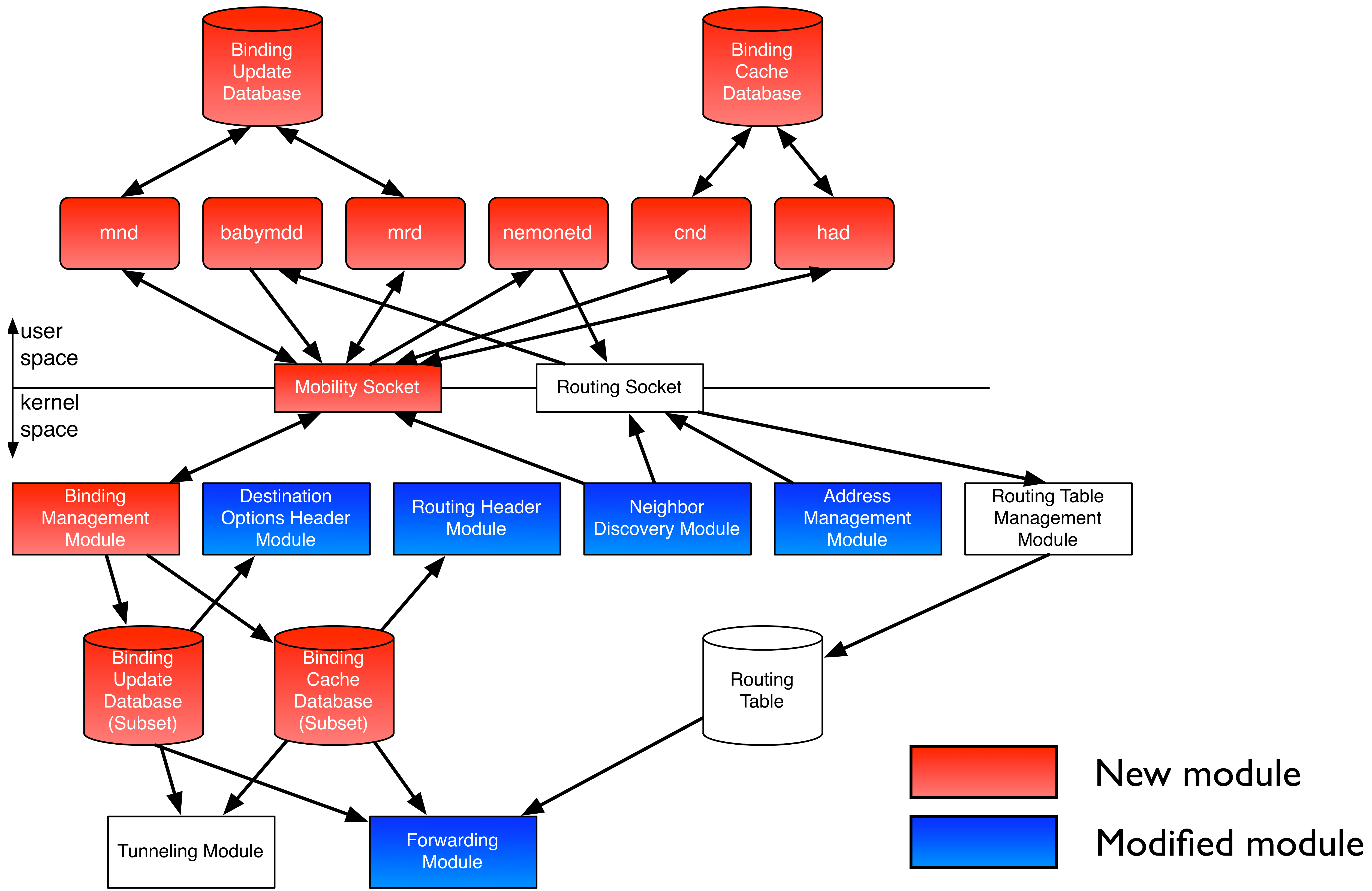
- SHISA consists of 6 programs and kernel

mnd	Mobile Host Functions
had	Home Agent Functions (for both Mobile IPv6 and NEMO BS)
cnd	Route Optimization Function
babymdd	A simple movement detector
mrd	Mobile Router Functions
nemonetd	Tunnel setup for NEMO BS
Kernel	Forwarding, tunneling processing

Node Configuration

- Selection of running programs decides the node type
- For Mobile Host
  - **mnd**, **babymdd** and **cnd** (if RO as a CN is required)
- For Home Agent
  - **had**, **cnd** (if RO as a CN is required) and **nemonetd** (if NEMO BS is required)

SHISA Modules



Mobility Socket

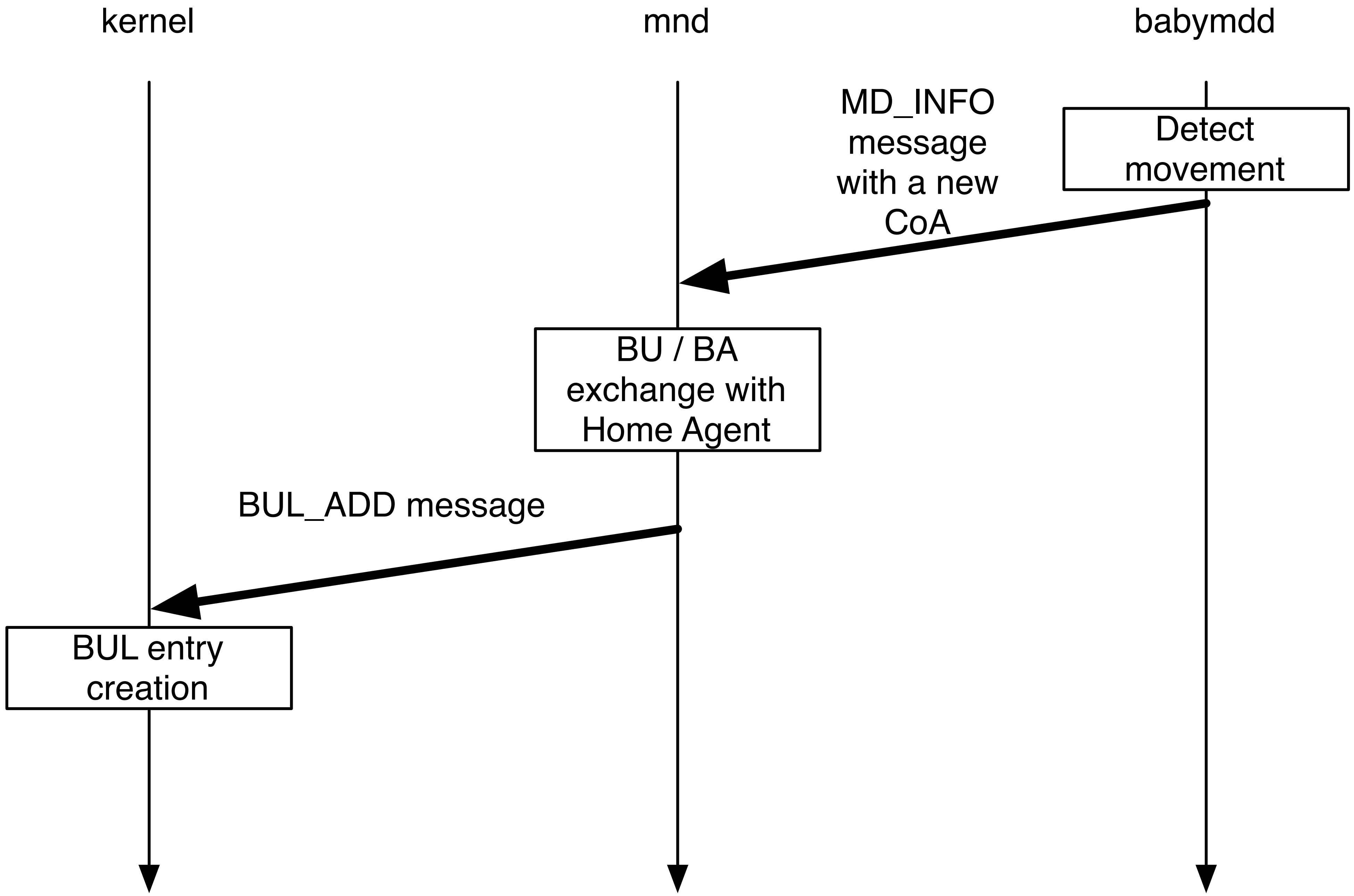
- A new communication domain socket (Mobility Socket, AF_MOBILITY) is designed
  - Similar to the Routing Socket
  - Address family independent (may be used with other mobility protocols)
- Mobility Socket provides
  1. Kernel interface to application programs
  2. Communication method between application programs

Mobility Socket Messages

NODETYPE_INFO	Configure the type of node (MN, MR, HA, CN)
BC_ADD	Add a Binding Cache entry
BC_REMOVE	Remove a Binding Cache entry
BC_FLUSH	Clear all Binding Cache entry
BUL_ADD	Add a Binding Update List entry
BUL_REMOVE	Remove a Binding Update List entry
BUL_FLUSH	Clear all Binding Update List entry
MD_INFO	Movement information
HOME_HINT	A hint message that a node returns home
RR_HINT	A hint message that a node receives a bi-directional tunneled packet
BE_HINT	A control message from kernel to send a Binding Error message
DAD	A control message to kernel to perform DAD for a specified address

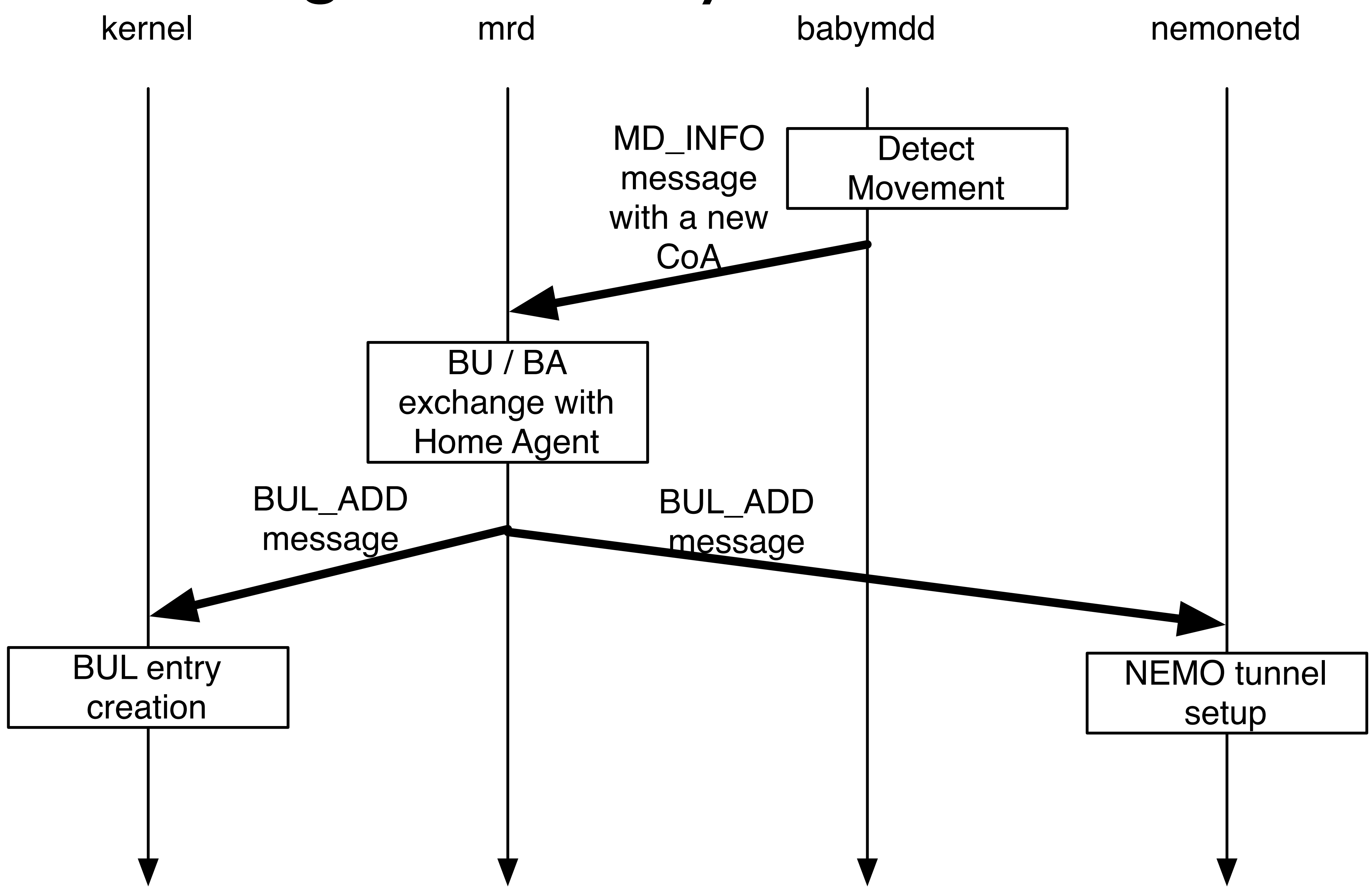
Message Passing Ex. 1

- Creating a Binding Update List entry



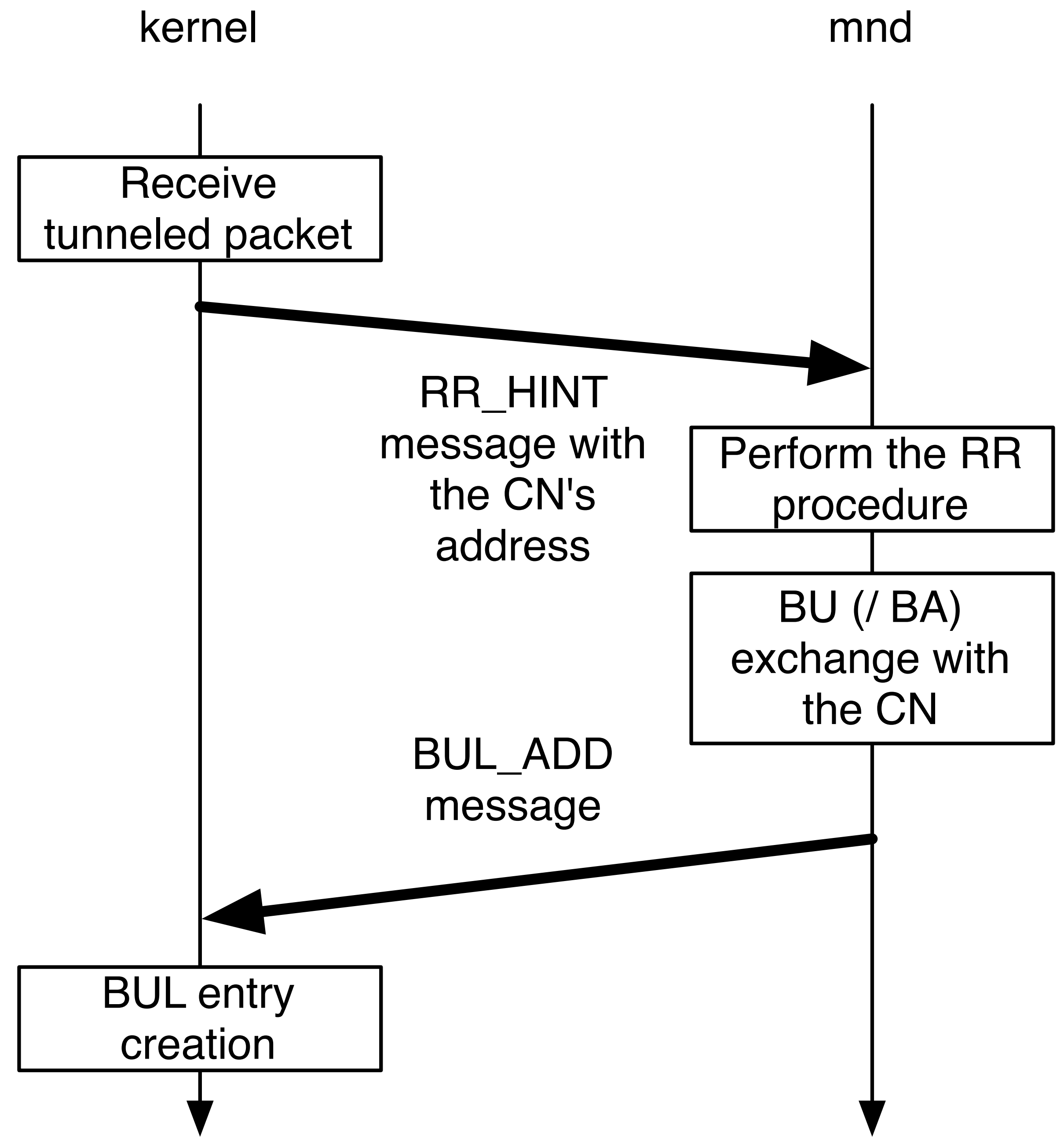
Message Passing Ex. 2

- Creating a BUL entry in the NEMO BS case



Message Passing Ex. 3

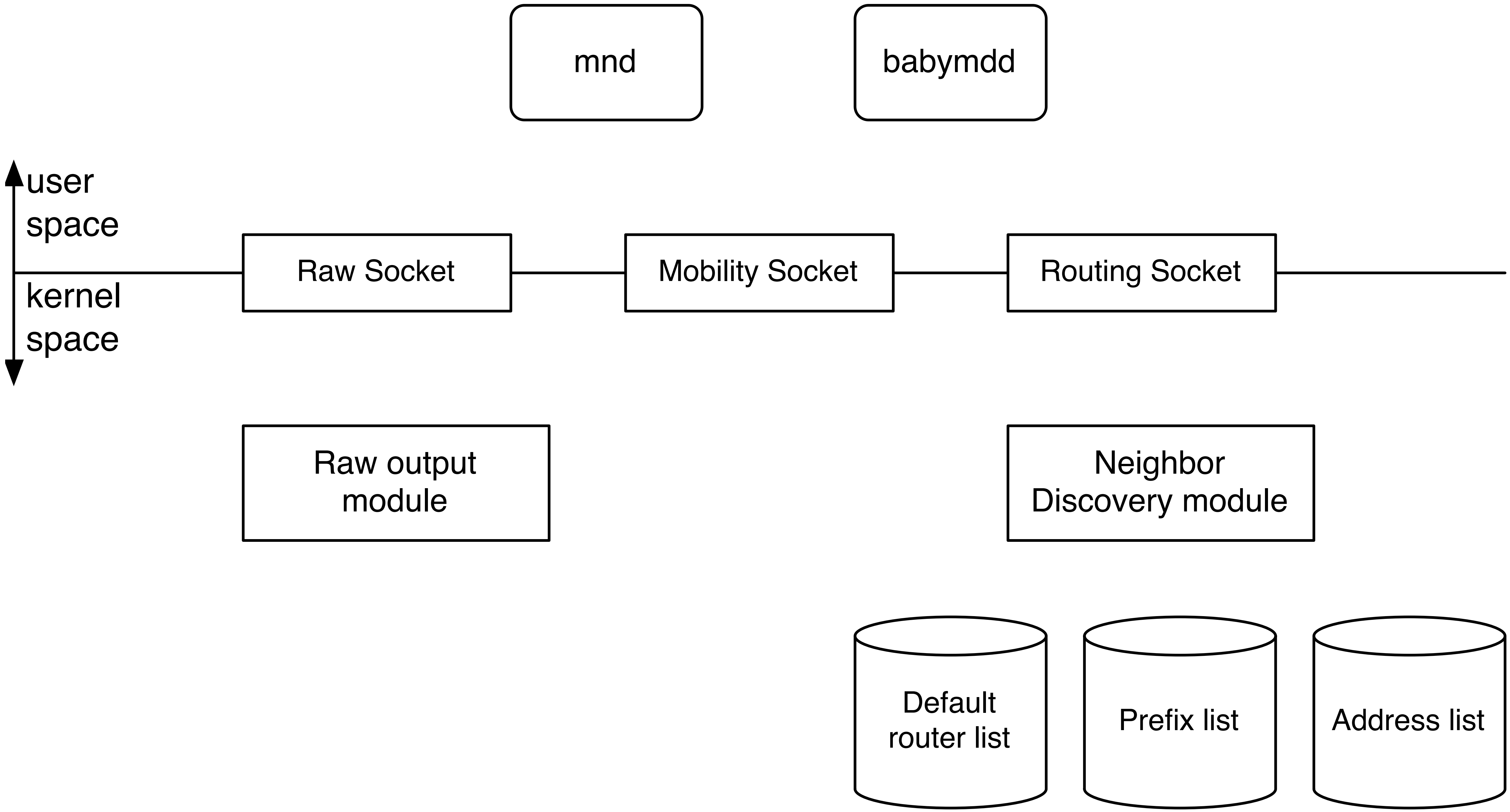
- Notification from the kernel



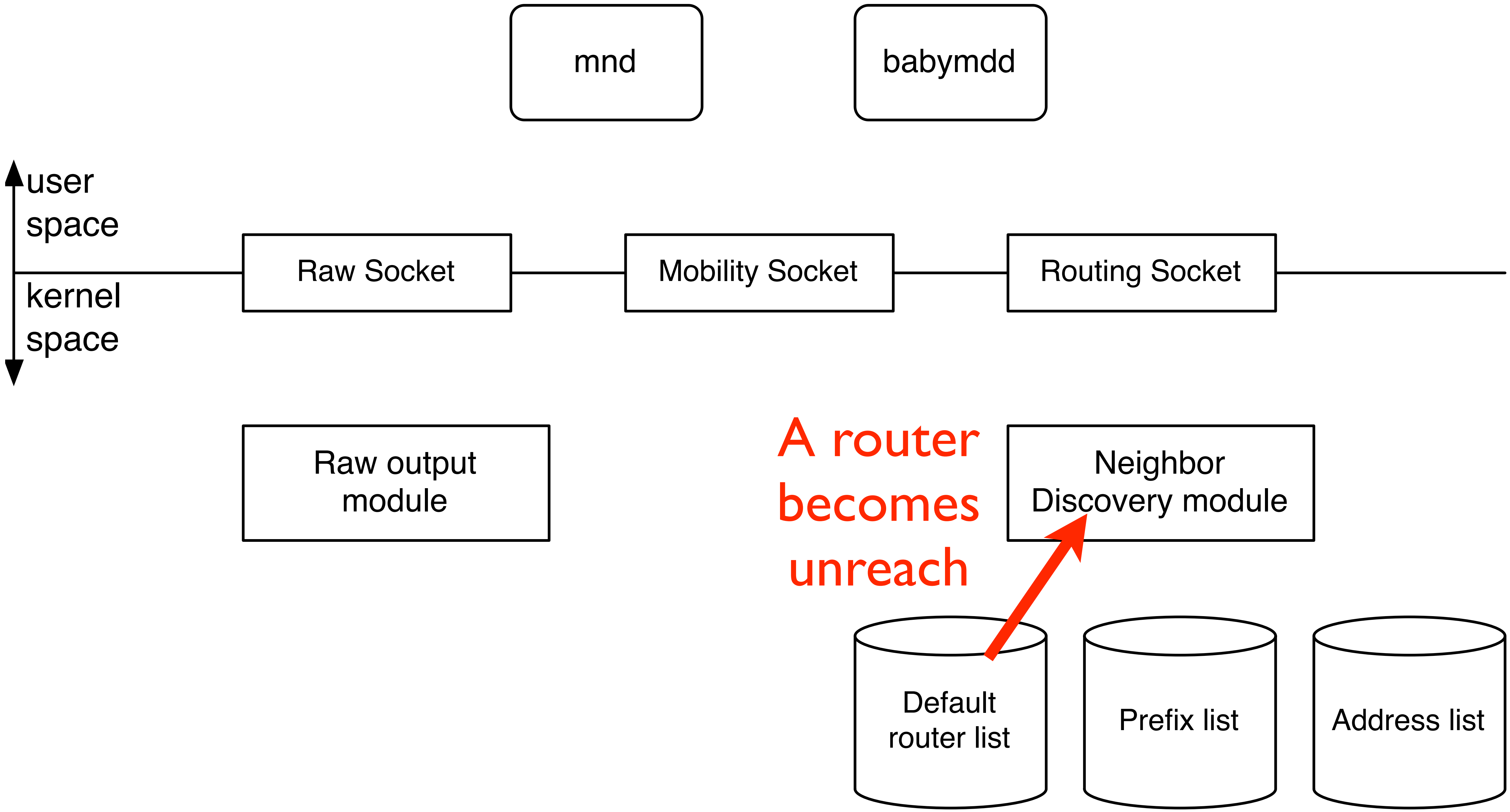
Movement Detection

- The **babymdd** program provides a basic movement detection function
- Based on the Neighbor Unreachability Detection (NUD)
  - When a router becomes unreachable, the prefixes advertised by the router becomes DETACHED state

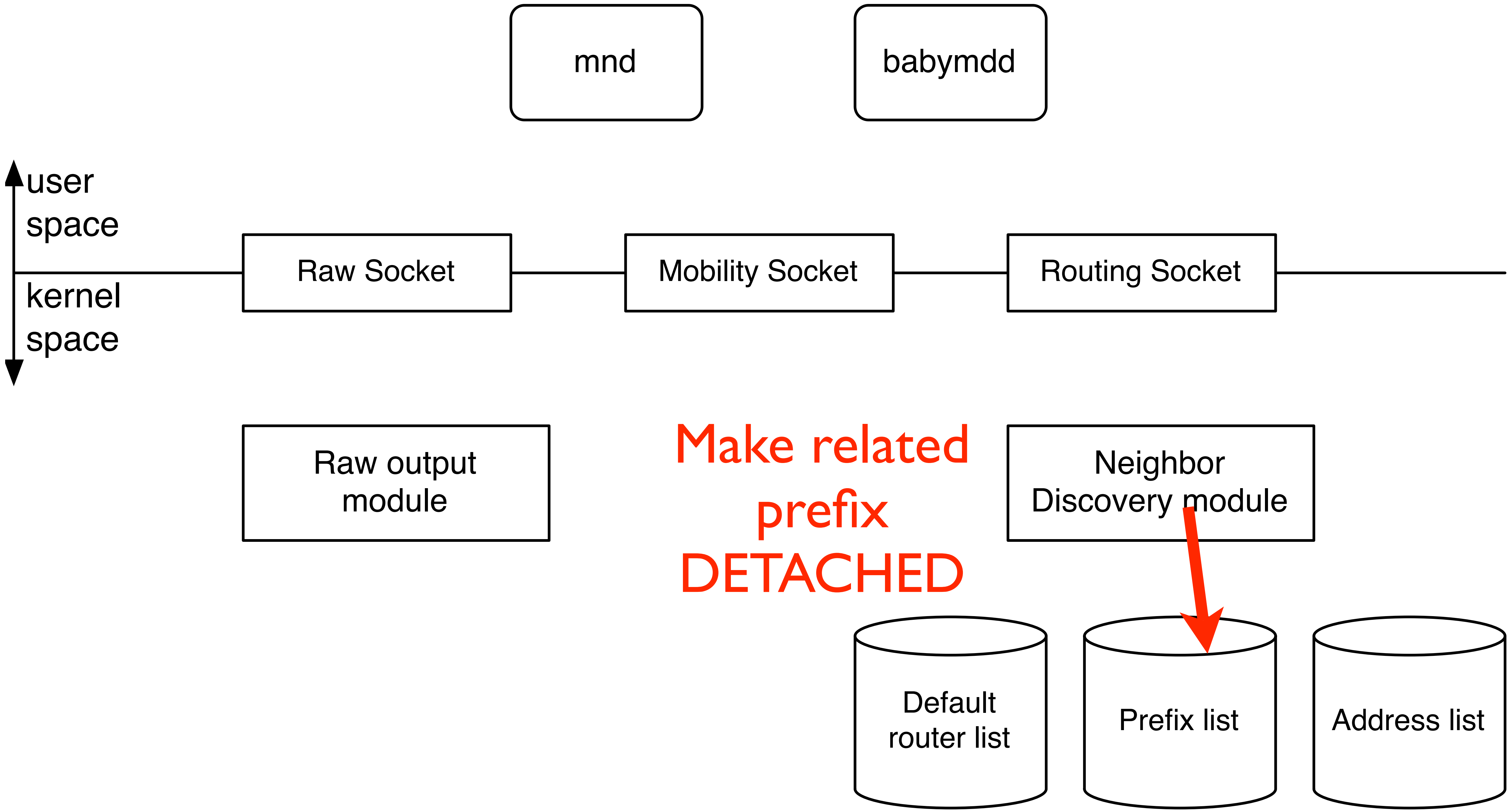
Movement Detection



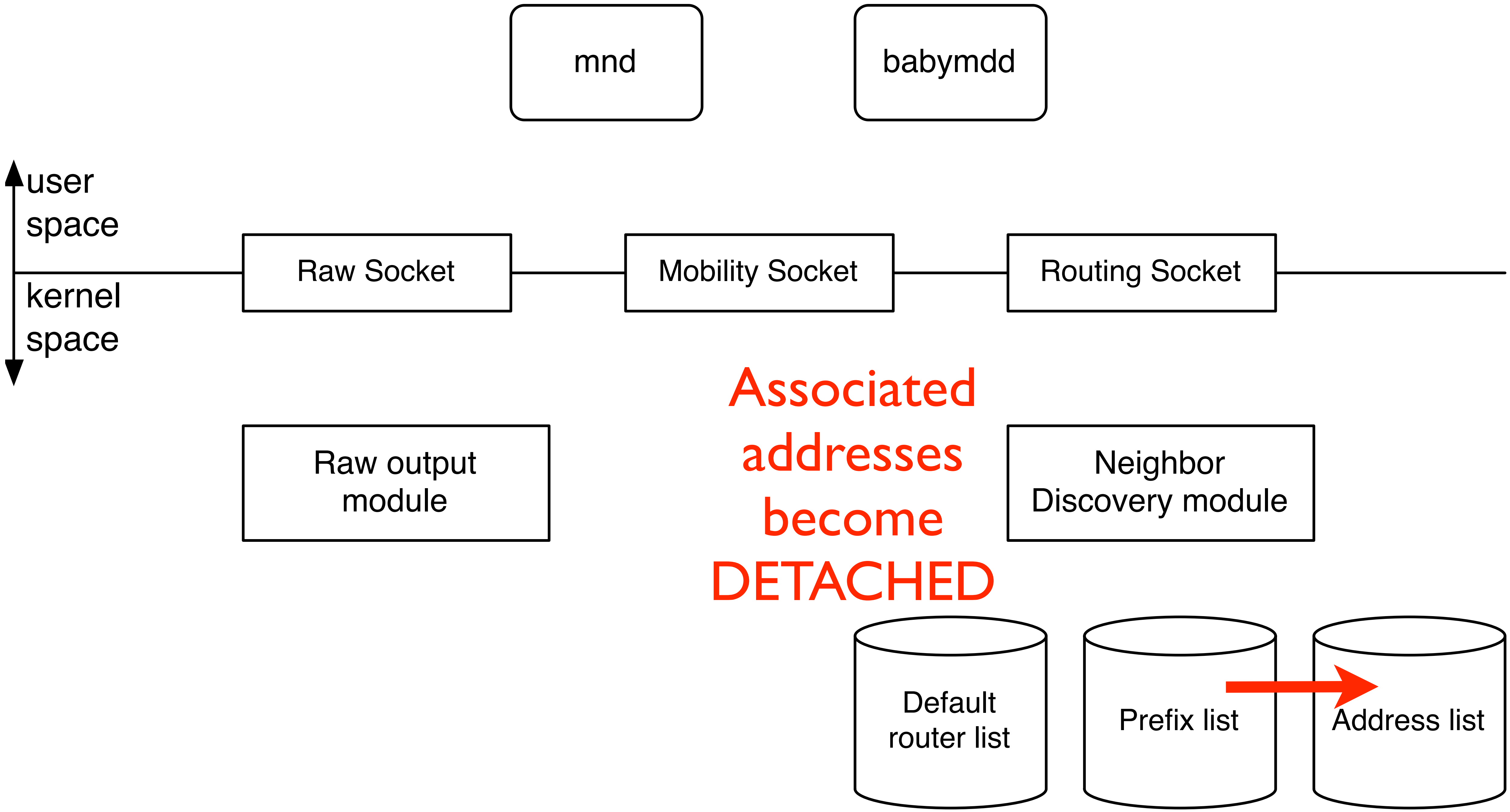
Movement Detection



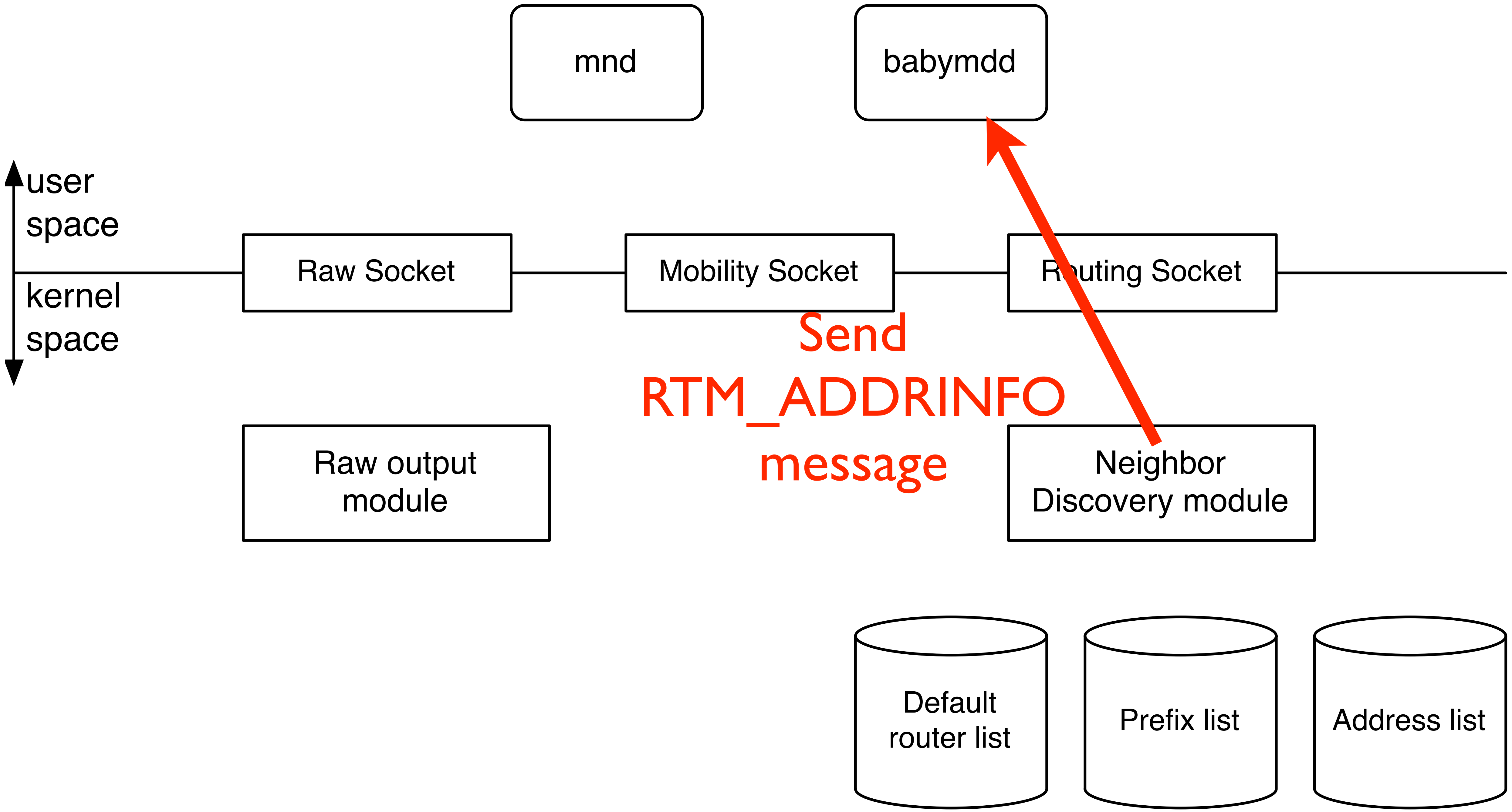
Movement Detection



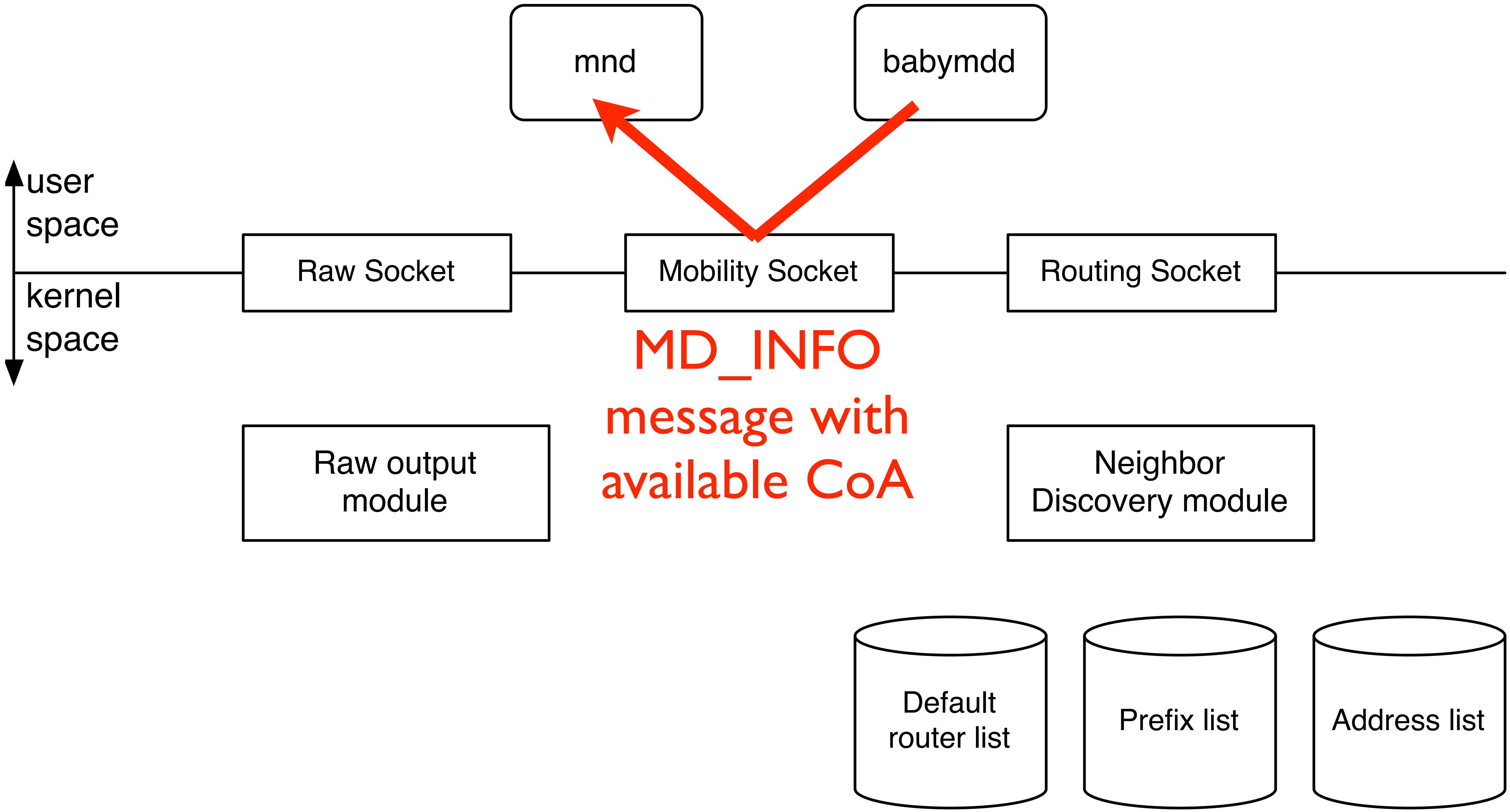
Movement Detection



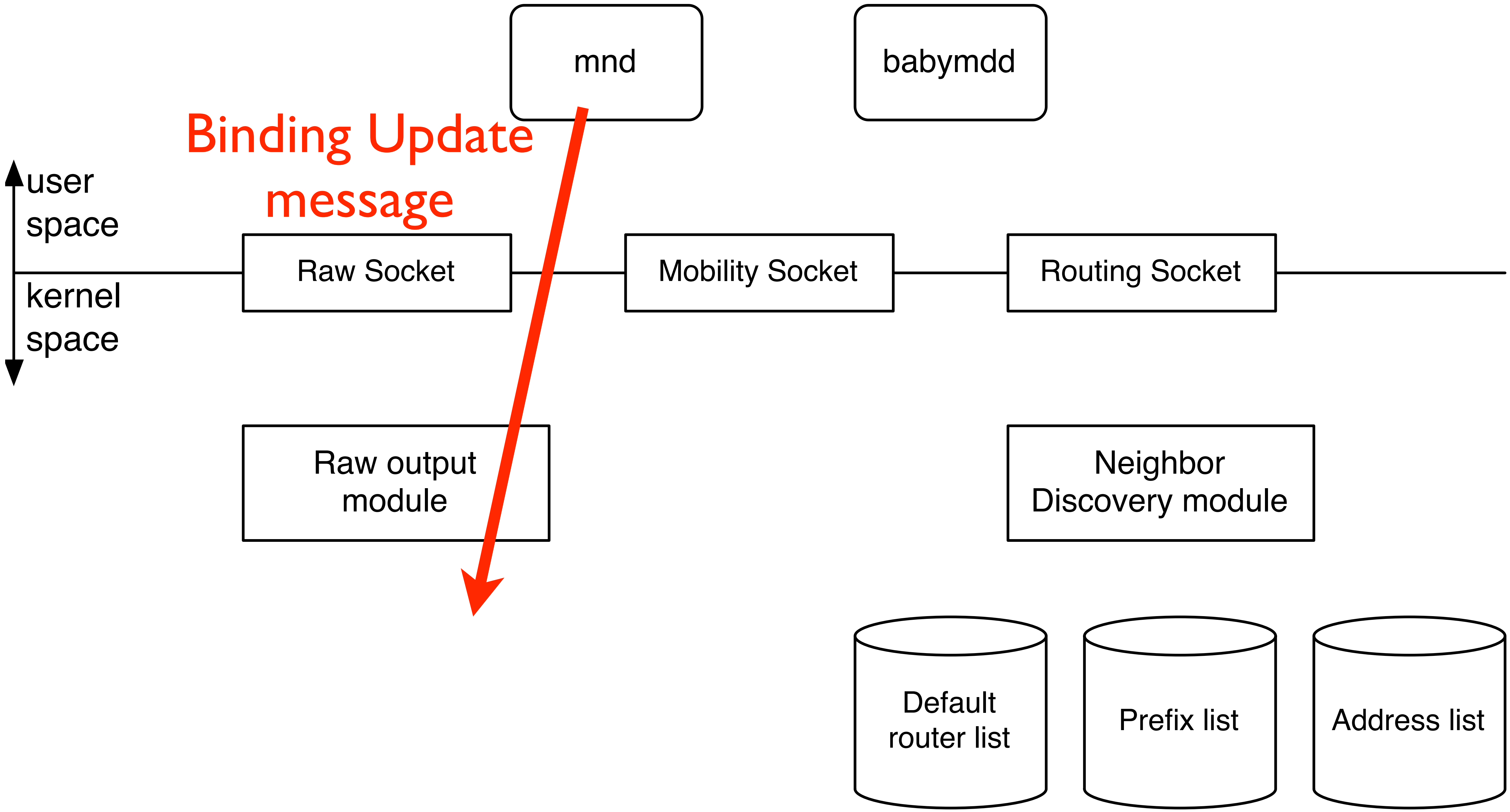
Movement Detection



Movement Detection



Movement Detection



Movement Detection

- The NUD mechanism is not necessarily utilized
- The requirement to send a Binding Update message is to send MD_INFO message
- Layer 2 aware, or some other special movement detection programs may enhance the handover performance

Address Flag Extension

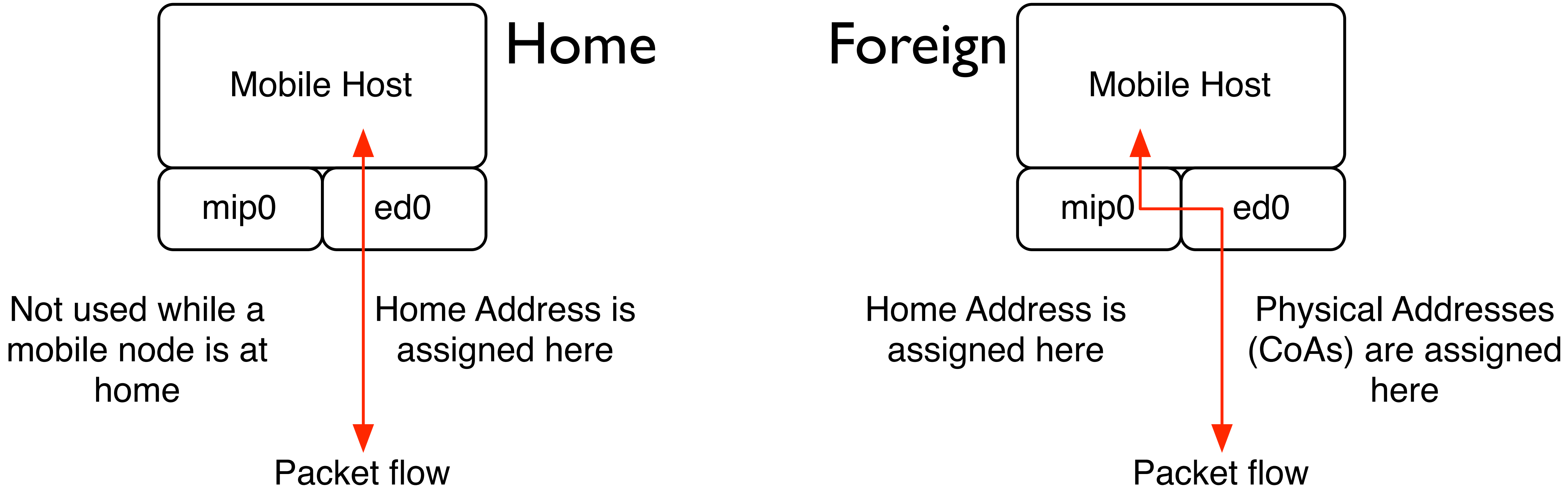
- `IN6_IFF_HOME`
  - Used with a home address to identify the address is a special address
  - Default Address Selection
- `IN6_IFF_DEREGISTERING` are added
  - Used with a home address to mark the address is not usable because of de-registration procedure

Pseudo Interface

- Home addresses are assigned to the physical interface attached to the home network while a mobile node is at home
- When the mobile node moves to a foreign network, the home addresses cannot stay there
  - The mip pseudo interface is defined as a placeholder of the home addresses and as a virtual home interface

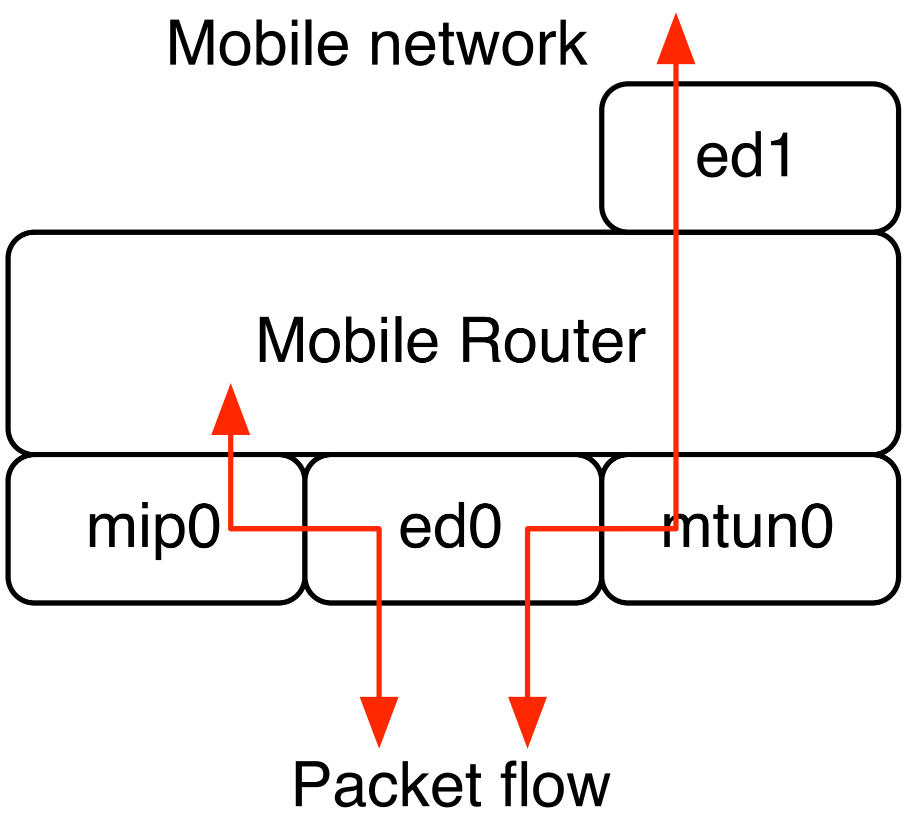
Pseudo Interface

- All packets sent from a mobile node is delivered to the mip interface and tunneled to the home agent of the mobile node
- Similar to the gif interface, but has been tightly integrated to mobility functions

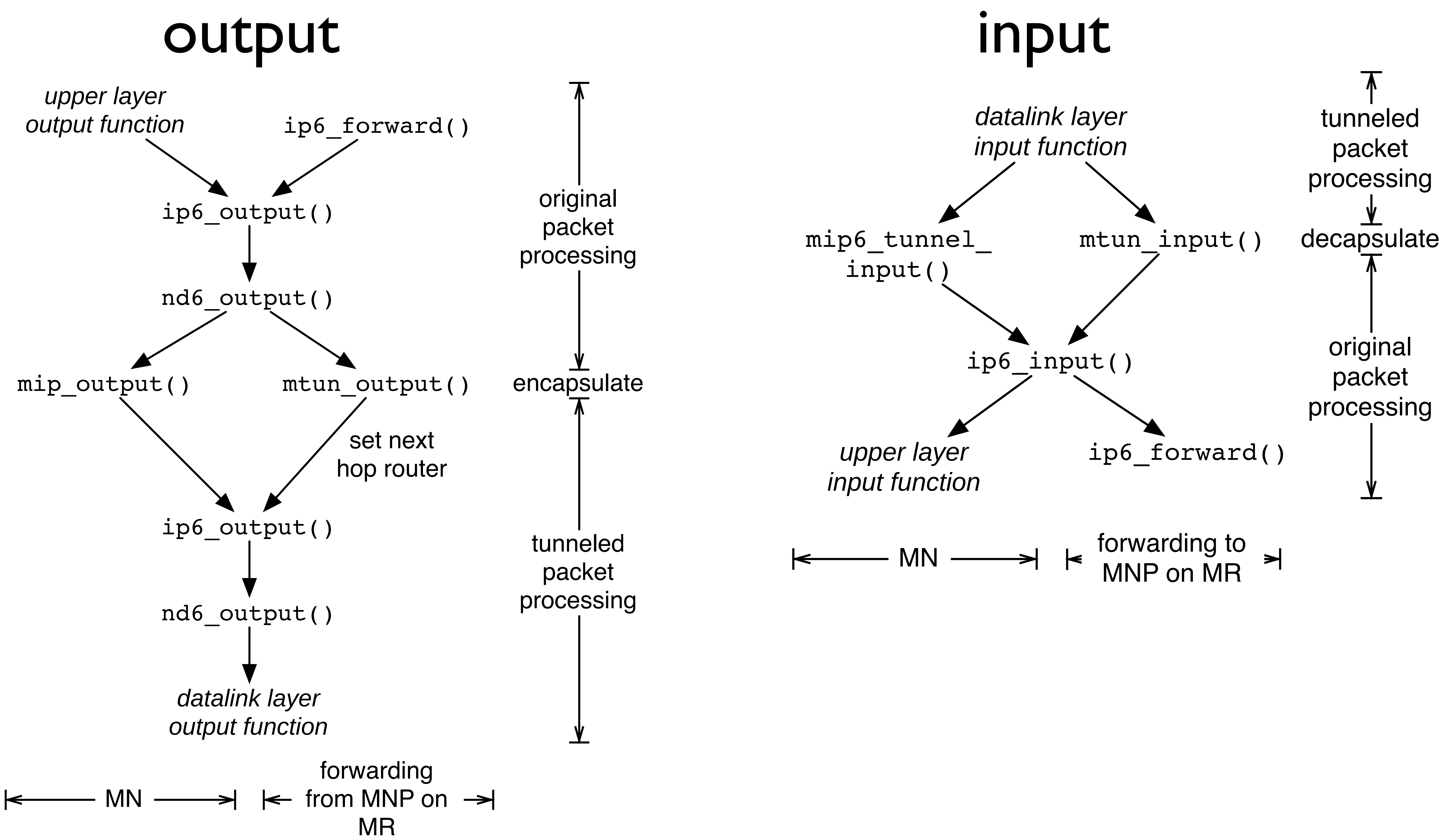


Pseudo Interface

- To support NEMO BS, we also use other pseudo interface (the mtun interface) for packet tunneling from/to nodes inside a mobile network



Tunneling Call Graph (on Mobile Node)



Mobility Header Messages

- Used to deliver IPv6 mobility signaling messages
- Defined as one of the extension headers in the specification (protocol number = 135)
  - However, in the current spec, the header is always the final header and never have following headers
- The kernel and application programs treat the header as a kind of transport header

Mobility Header Messages

- Implemented as one of the raw sockets
  - The `inet6sw[]` instance is extended to support this new header
- The input routine simply validates the incoming messages and passes them to raw sockets
- Application programs can send the messages using raw sockets

Consideration

- Duplicated mobility tunneling mechanisms
- IPsec policy management
- Mobility message passing mechanism unification
- IKE interaction
- Porting to other systems

Duplicated Mobility Tunnel

- The tunneling mechanism for Mobile *Host* traffic and that for Mobile *Network* are implemented separately
- It is too confusing and we are now planning to integrate these tunneling
- You may wonder why we don't use the gif interface
  - We need many mobility related processing in the tunnel packet processing, and it is not a good idea to extend the gif interface to do the jobs

IPsec Policy Management

- As defined in RFC, mobility signaling messages must be protected by the IPsec mechanism
- But the policy is different based on the location of a mobile node
  - e.g. HoT/HoTI messages
- Need to modify the policy entries dynamically

Mobility Message API

- We defined the Mobility Socket to exchange mobility related information between kernel and applications, or between applications
- Such interface may be useful for other mobile aware applications
- Also, it may increase portability of the mobility protocol processing applications
    - e.g. Porting SHISA to other OSes
- We may need to standardize the API

IKE Interaction

- Configuring IPsec SA entries is not an easy task, especially we manage many nodes
- Dynamic SA creation is necessary when we think real deployment scenarios
- We are not working with the Racoon2 development team to provide IKE integration to SHISA

Porting to Other BSDs

- The original SHISA (developed in the KAME project) supported FreeBSD5.4R and NetBSD2.0 (and partially OpenBSD3.0)
- Currently we are concentrating our resource to port the developed code to NetBSD-current
- Once we have completed the integration to a certain level, we will start working on FreeBSD, and OpenBSD later
- (BTW, although it is unofficial, Tsuyoshi Momose who is one of SHISA developers is porting SHISA to Darwin)

Conclusion

- Designed a mobility stack with the following characteristics
  - Signal/Data processing separation for easy development
  - Adaptive movement detection mechanism
  - Simple mobility application interface
  - Small kernel modification
- Implemented the stack to satisfy the above requirements
- The code is freely available from the KAME project
- Now we are working to integrate the developed code to integrate to the BSD operating systems

Thank you!

Any Questions?