The Case for a Unified Extensible Data-centric Mobility Infrastructure

Yun Mao, Boon Thau Loo
Zachary Ives, Jonathan M. Smith
University of Pennsylvania

Aug 27, 2007
Mobiarch, Kyoto, Japan

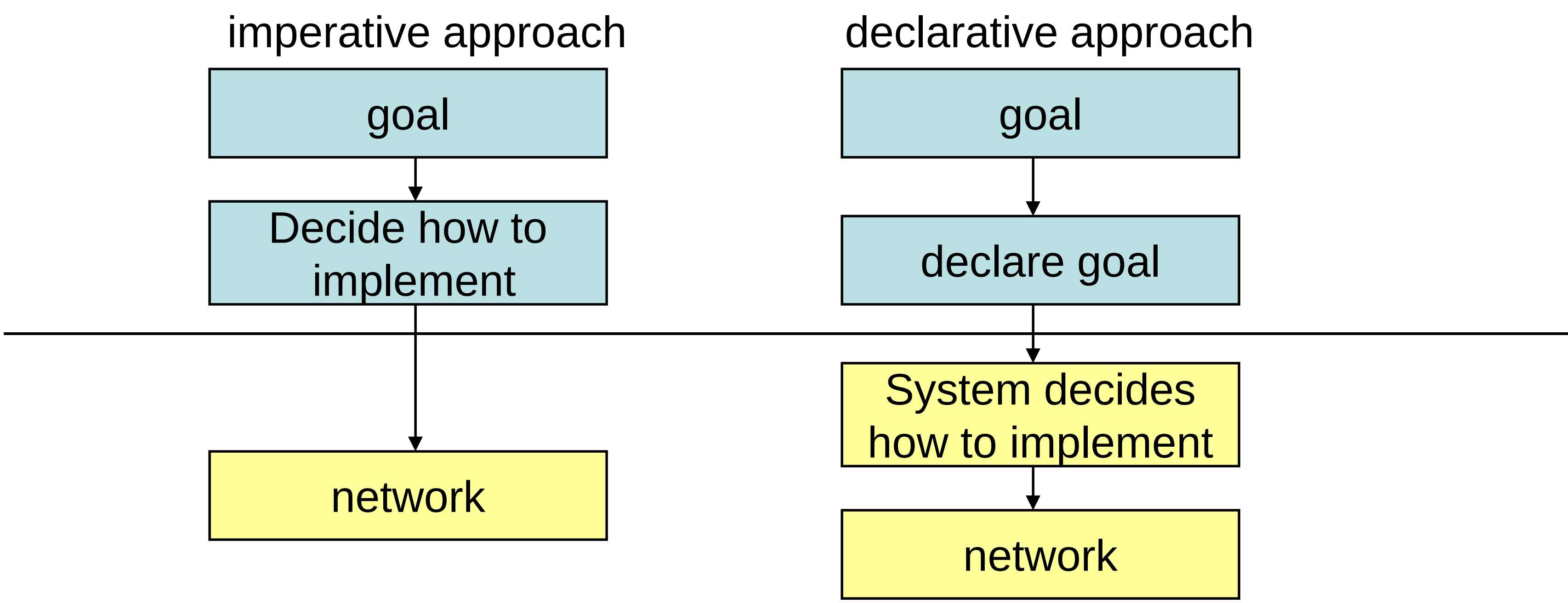
Project funded by NSF NeTS-0721845

Motivation

- Mobile environment is the driving force of the Internet architecture redesign
  - New environment, diversified wireless technologies
  - New applications and services
    - Naming, location-aware, context-aware services, etc
- Current network architecture is not extensible to meet the demand
  - Slow adoption of Mobile IP, IPv6
  - Too many application specific overlay networks
  - Emerging networks: sensor, DTN
- Problem: the architecture is tightly coupled with implementation

Our approach: a declarative architecture

- Core idea: separate logical specification from physical implementation:
  - "Ask for what you want, not how to implement it"
  - SQL vs C in the DB community in the 70s

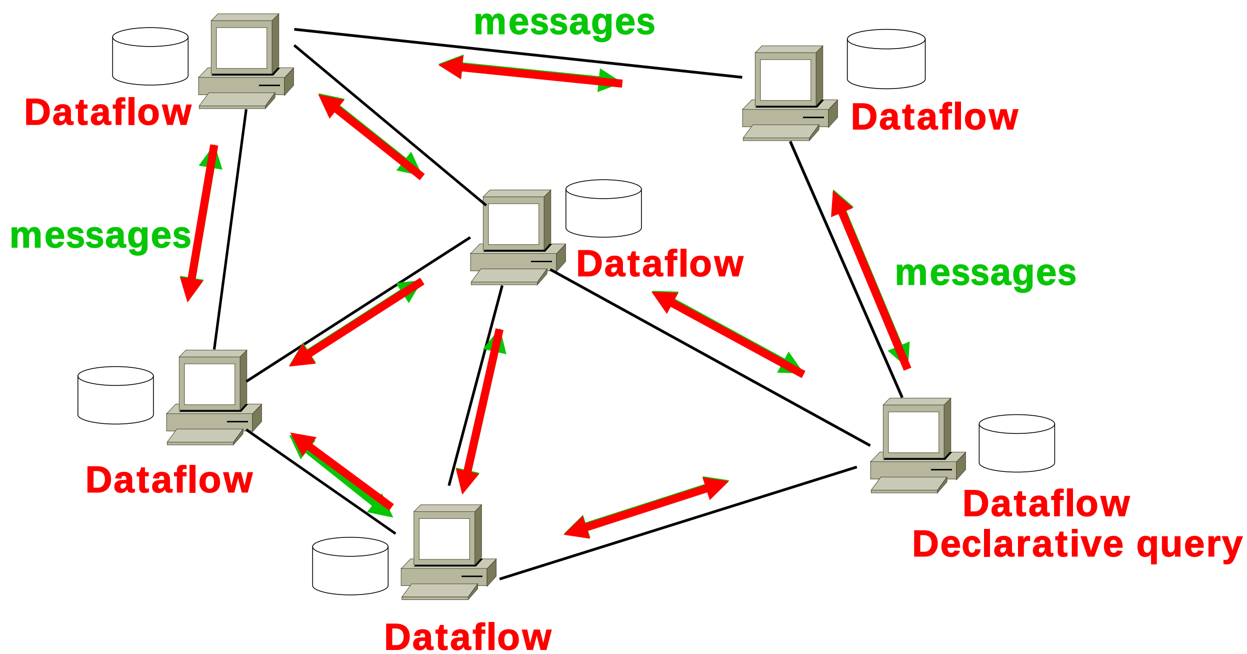


What is the right high-level abstraction in mobile networks?

- Network state as distributed data
  - Router state
  - Mobile home agent state
  - Service description
  - Network load
- Cross-layer info as data
  - All kinds of link/physical layer state
  - Session state
  - Application state
- Network protocols: the implementation of querying/updating data across domains/hosts and layers.

Declarative Networks - *data-centric* approach to networking

(SIGCOMM'05)



Traditional Networks	**Declarative Networks**
Network State	Distributed db tables
Network protocol	Declarative Query Execution
Network messages	Distributed Dataflow

Datalog: a declarative language

Datalog rule syntax:

<result> ← <condition1>, <condition2>, ... , <conditionN>.

Head **Body**

- Similar to Prolog
- Types of conditions in body:
  - Input tables: *link(src,dst)* predicate
  - Arithmetic and list operations
- Head is an output table
  - Recursive rules: result of head in rule body

Routing as a Query

- R1: reachable(S,D) ← link(S,D)
- R2: reachable(S,D) ← link(S,Z), reachable(Z,D)

“For all nodes S,D,

***link(a,b)* – “there is a link from node *a* to node *b*”**

If there is a link from S to D, then S can reach D”.

***reachable(a,b)* – “node *a* can reach node *b*”**

“For all nodes S,D, if there is a link from S to Z,

and there is a path from Z to D, then S can reach D”.

- Input: link(source, destination)
- Output: reachable(source, destination)

Network Datalog (NDLog)

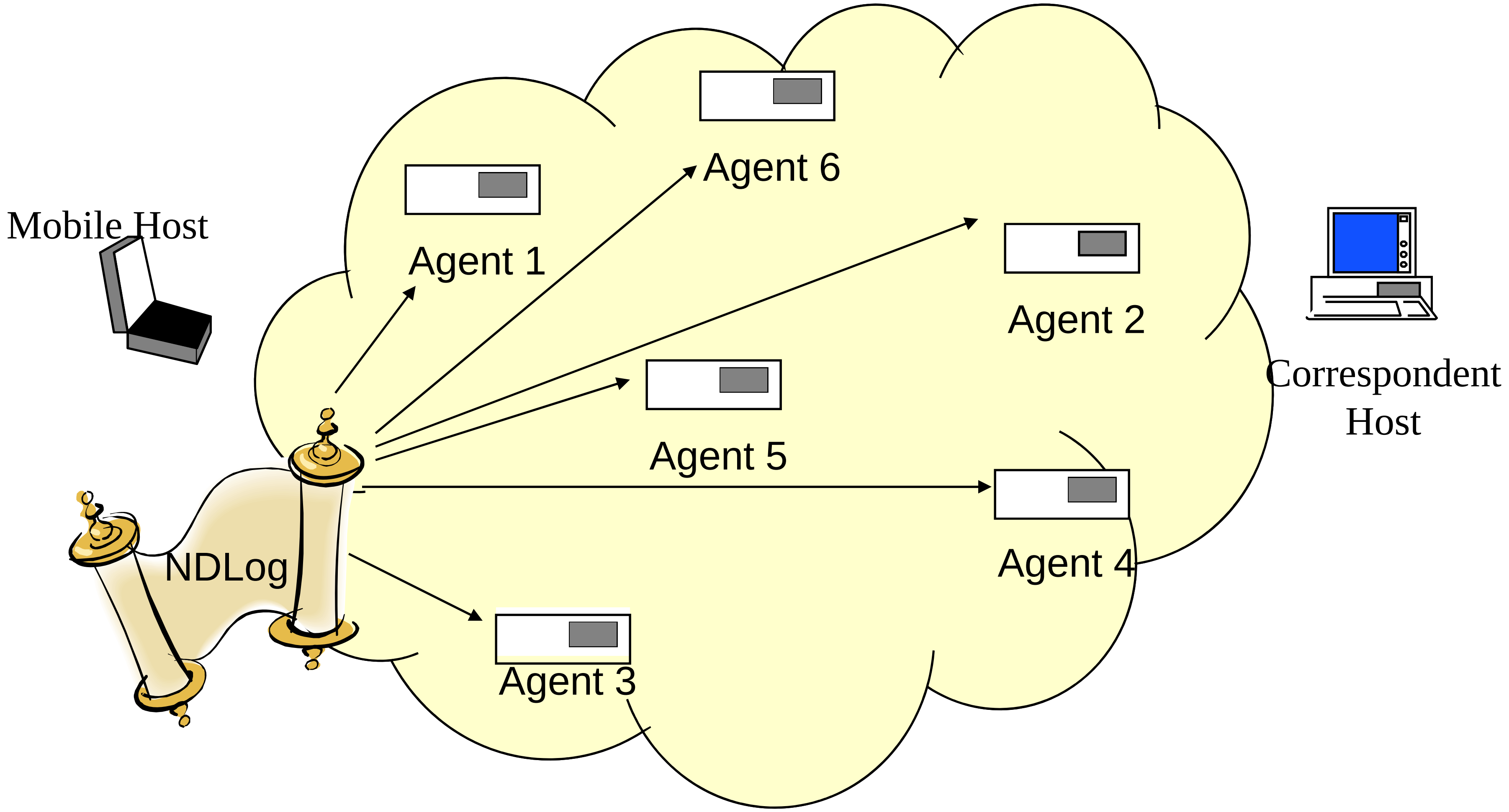
- A distributed variant of Datalog
  - express distributed computation over network state
  - Compiled into distributed data flows, and executed by query processors (i.e. routers)
- Concise yet expressive:
  - Textbook routing protocols (3-8 lines)
    - Distance-vector, dynamic source routing, path-vector
    - QoS routing
  - Chord DHT (48 lines)
- Efficiency

NDLog Examples in Mobile Networks

- Mobile home agent selection for overlays
  - i3 / Roam [MobiSys'03] (16 lines)
  - DHARMA [Infocom'05] (10 lines)
- Naming
- Customizable routing
- Service discovery and composition

```
T1 leastLoad(@PI, SI, min<L>) :- proxy(@SI, P, PI), transcoders(@PI,  TI, TID, L).
T2 bestTranscoder(@SI, TI, TID) :- transcoders(@PI, TI, TID, L), leastLoad(@PI, SI, L).
Query bestTranscoder(@SI, TI, TID).
```

Declarative Mobile Networking (at 30000 feet)

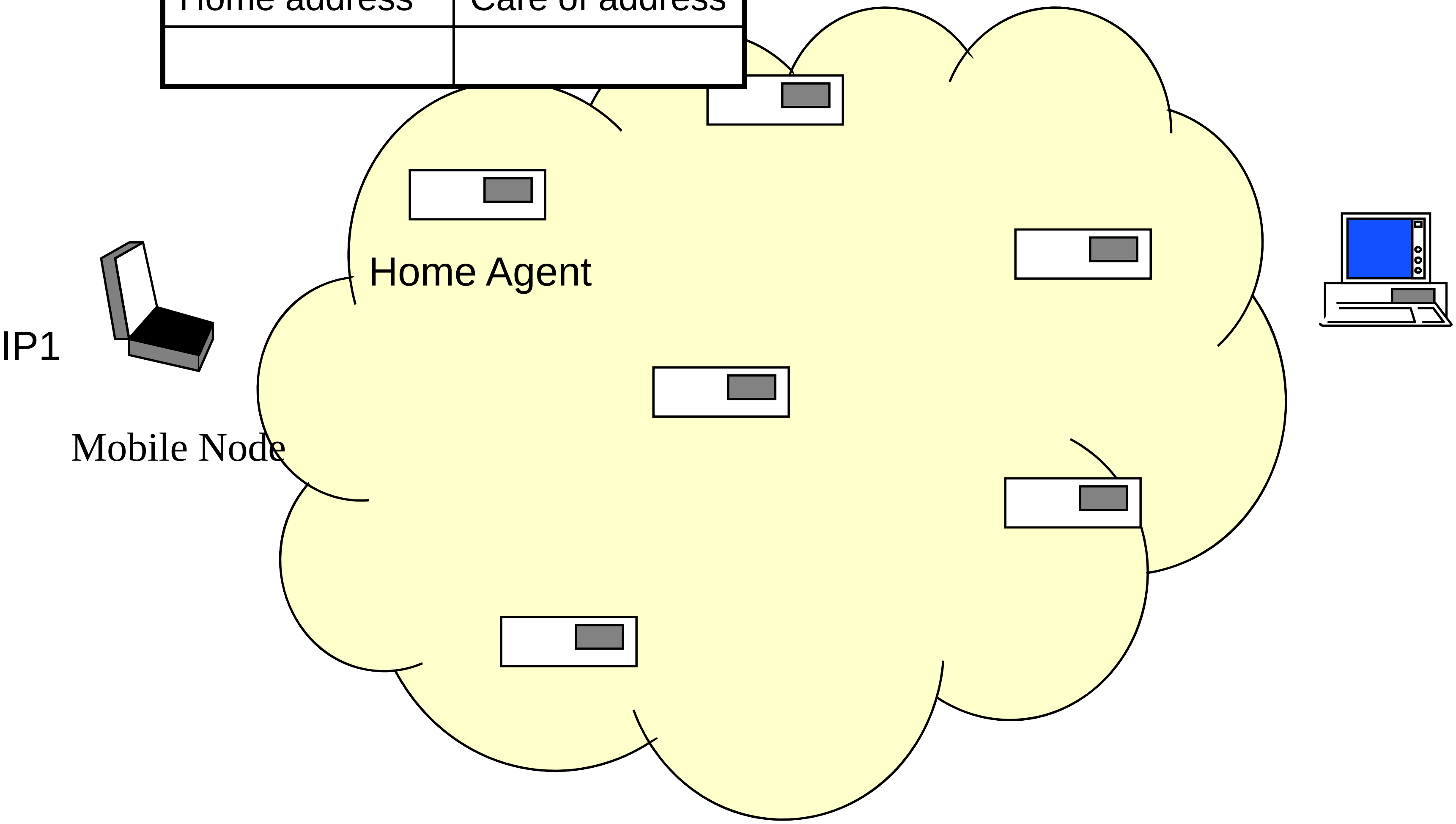


Out-of-band deployment of distributed queries by ISPs

Mobile IP

Mobility binding table

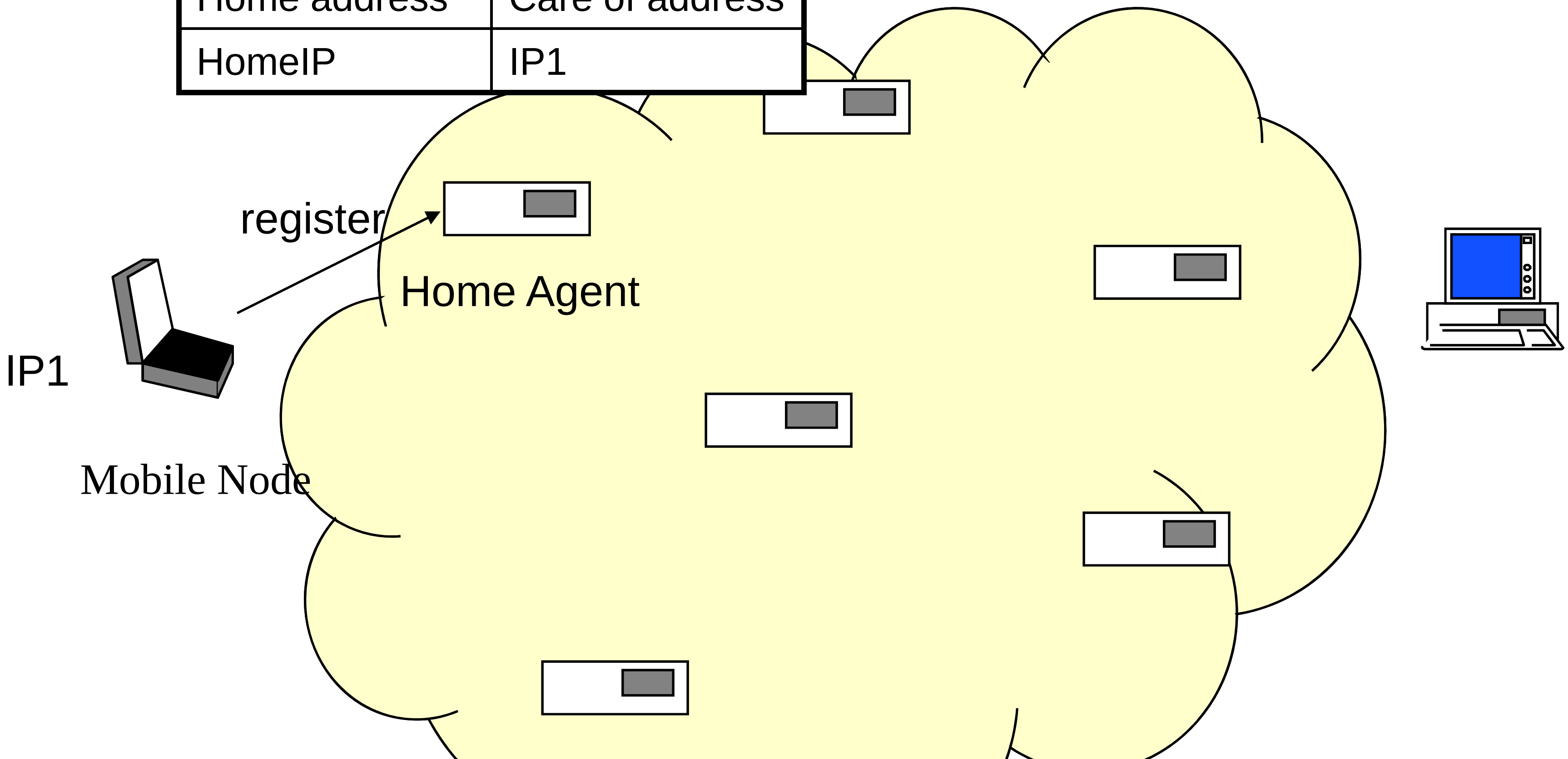
Home address	Care of address



Mobile IP

Mobility binding table

Home address	Care of address
HomeIP	IP1

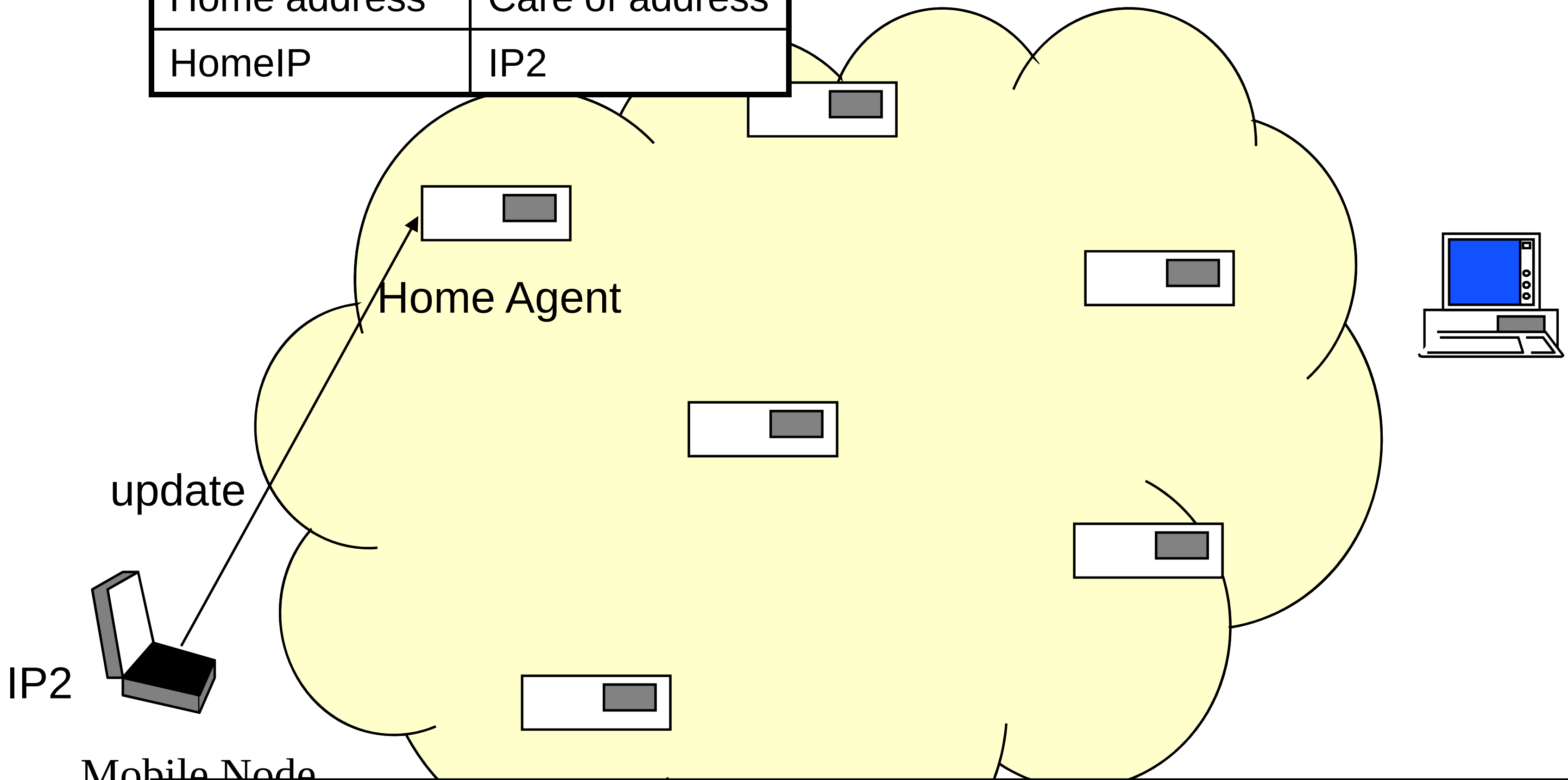



```
mobile_ip_binding(@HA, HomeIP, CurrentIP)
    :- ip_change_event(@MN, CurrentIP), homeip(@MN, HomeIP)
```

Mobile IP

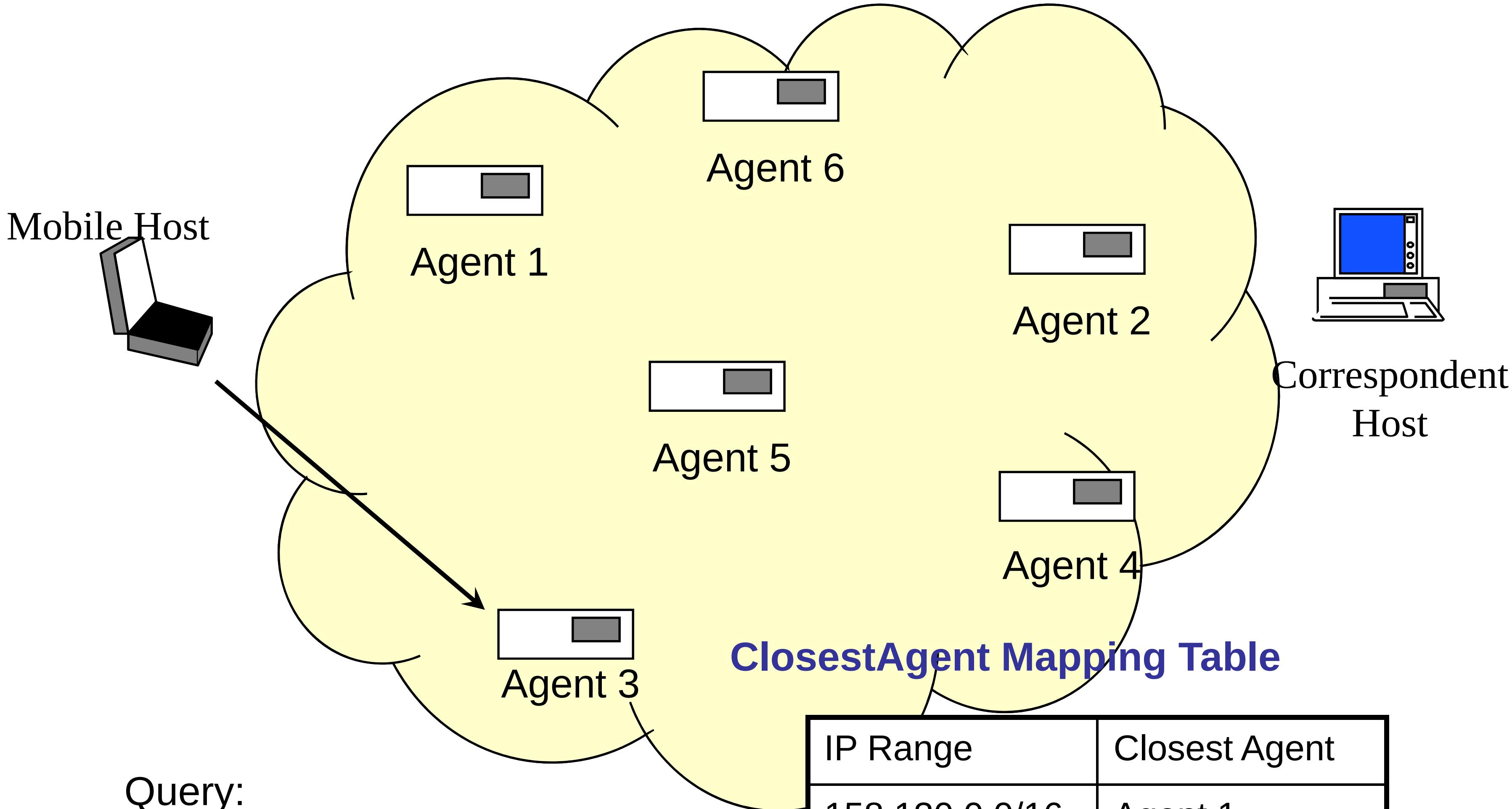
Mobility binding table

Home address	Care of address
HomeIP	IP2



mobile_ip_binding(@HA, HomeIP, CurrentIP)
    :- ip_change_event(@MN, CurrentIP), homeip(@MN, HomeIP)

Declarative HA Selection

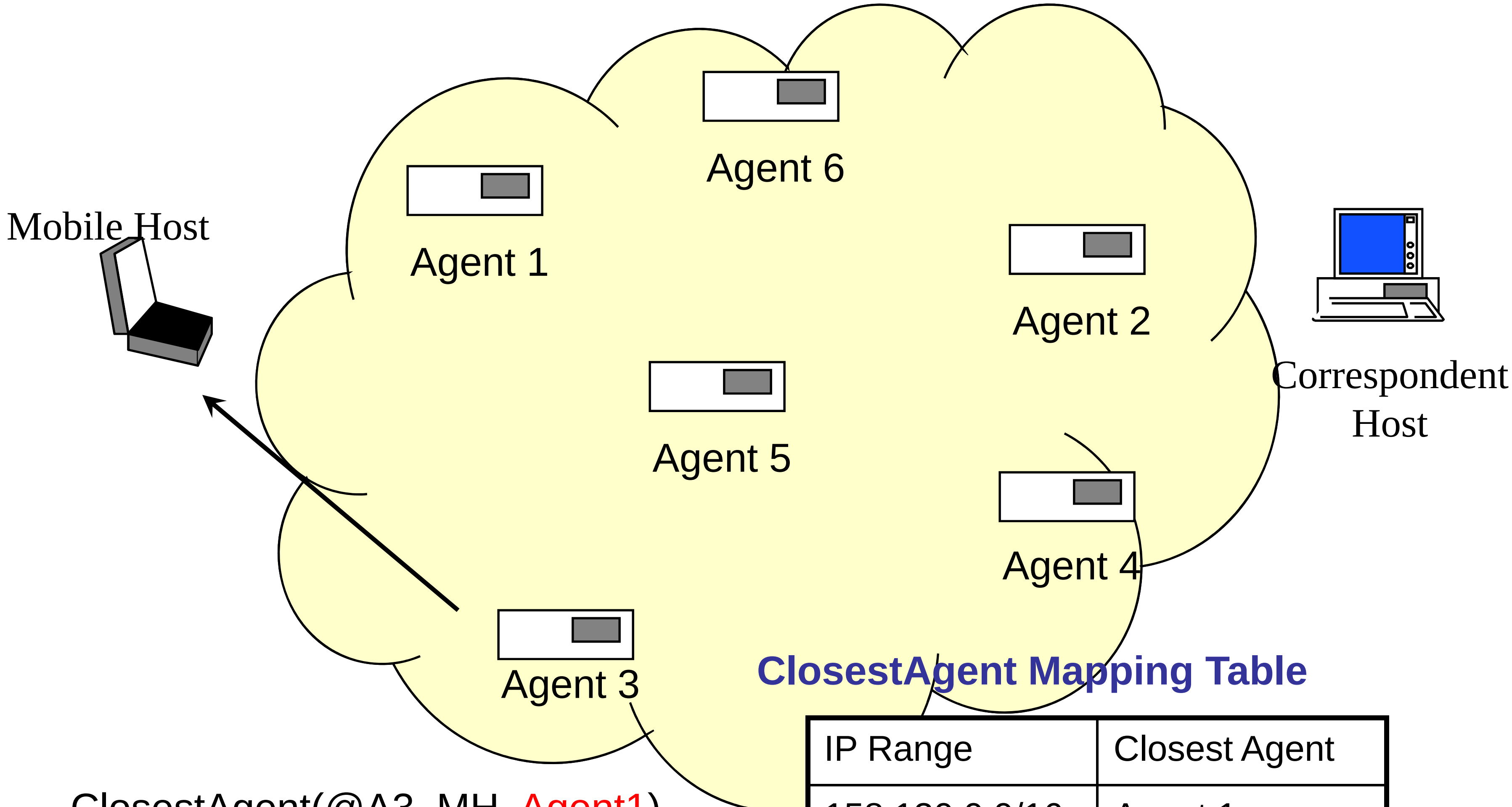


ClosestAgent Mapping Table

IP Range	Closest Agent
158.130.0.0/16	Agent 1
….	Agent 2

Query:
ClosestAgent(@A3, MH, HA)

Declarative HA Selection



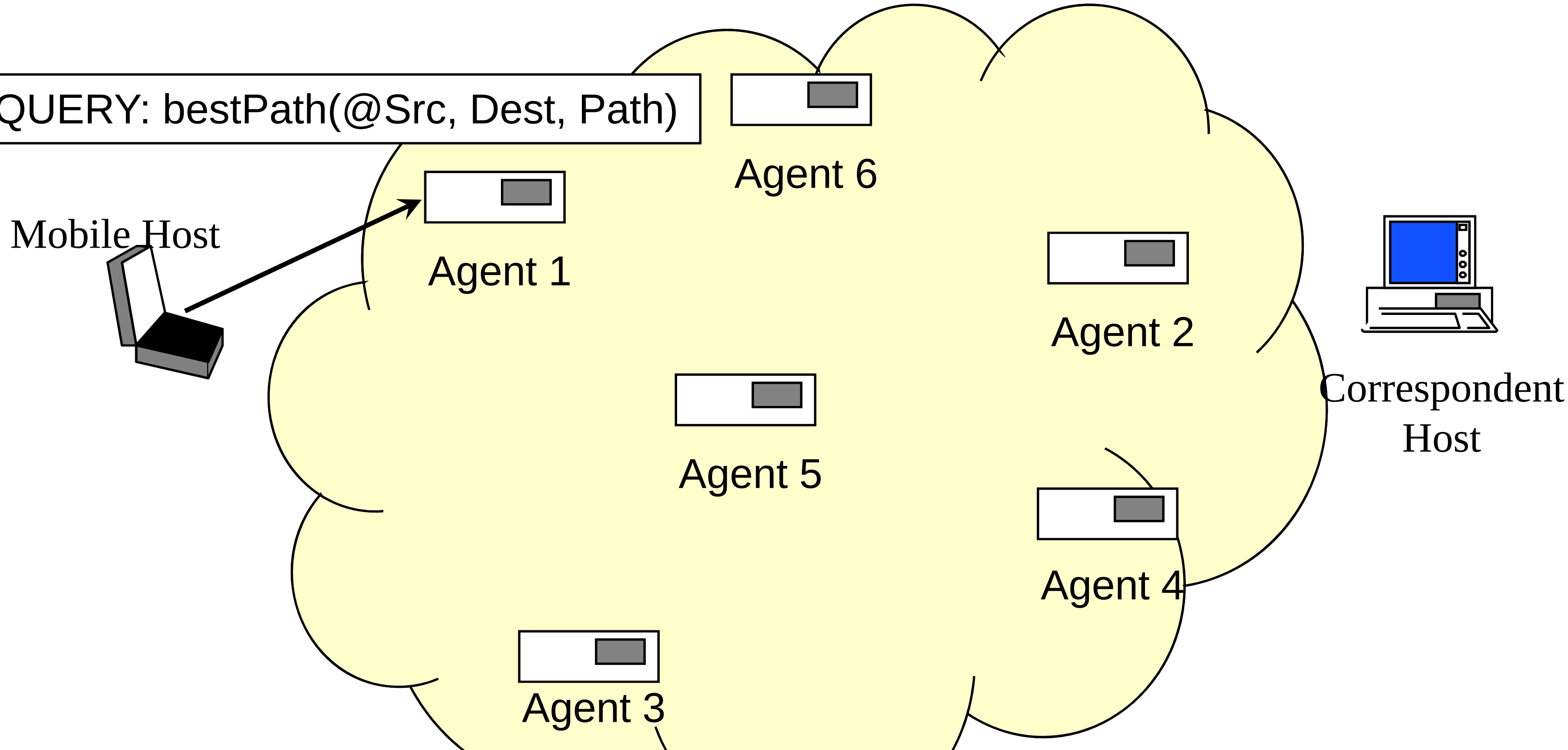
ClosestAgent(@A3, MH, Agent1)

ClosestAgent Mapping Table

IP Range	Closest Agent
158.130.0.0/16	Agent 1
….	Agent 2

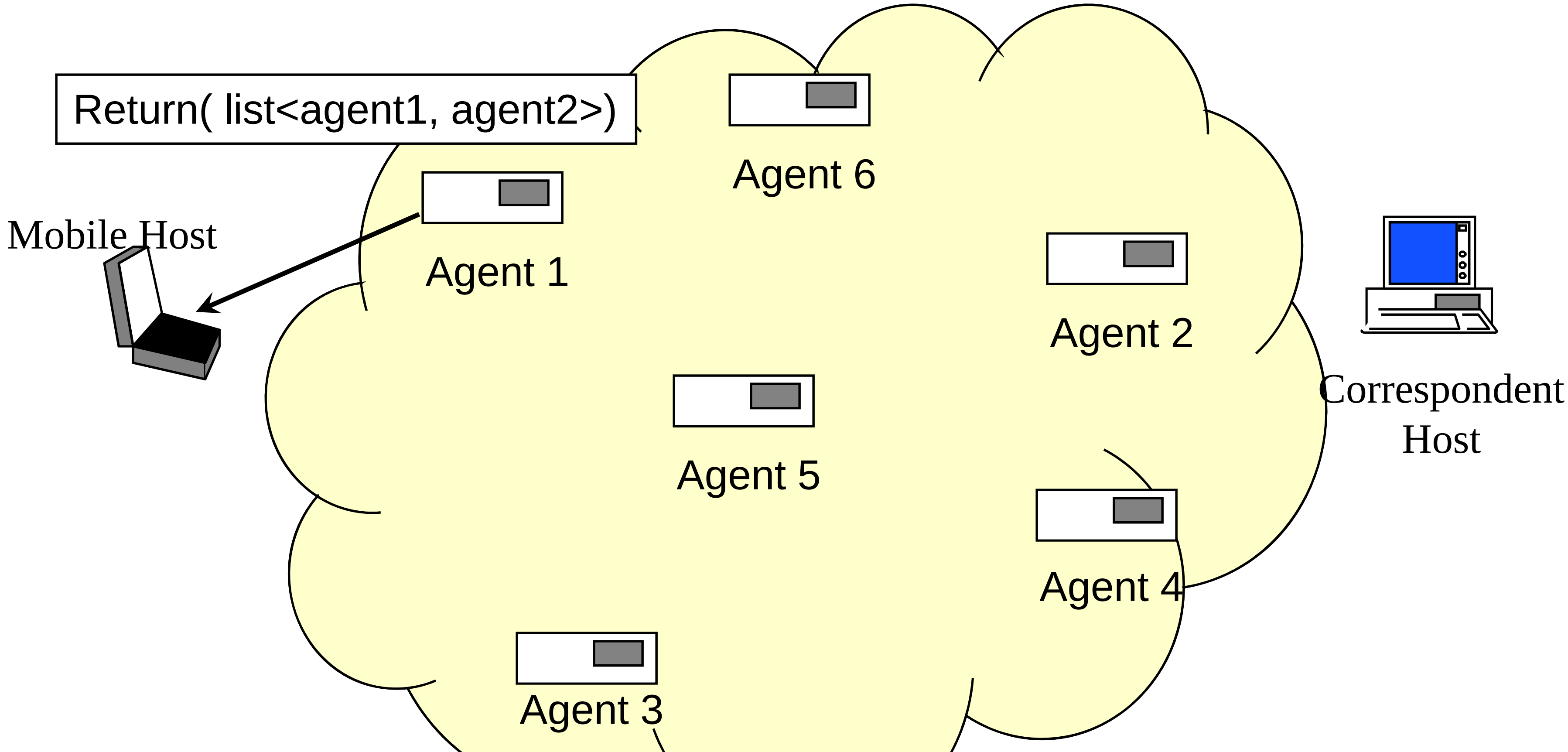
QoS-aware routing: low latency

QUERY: bestPath(@Src, Dest, Path)



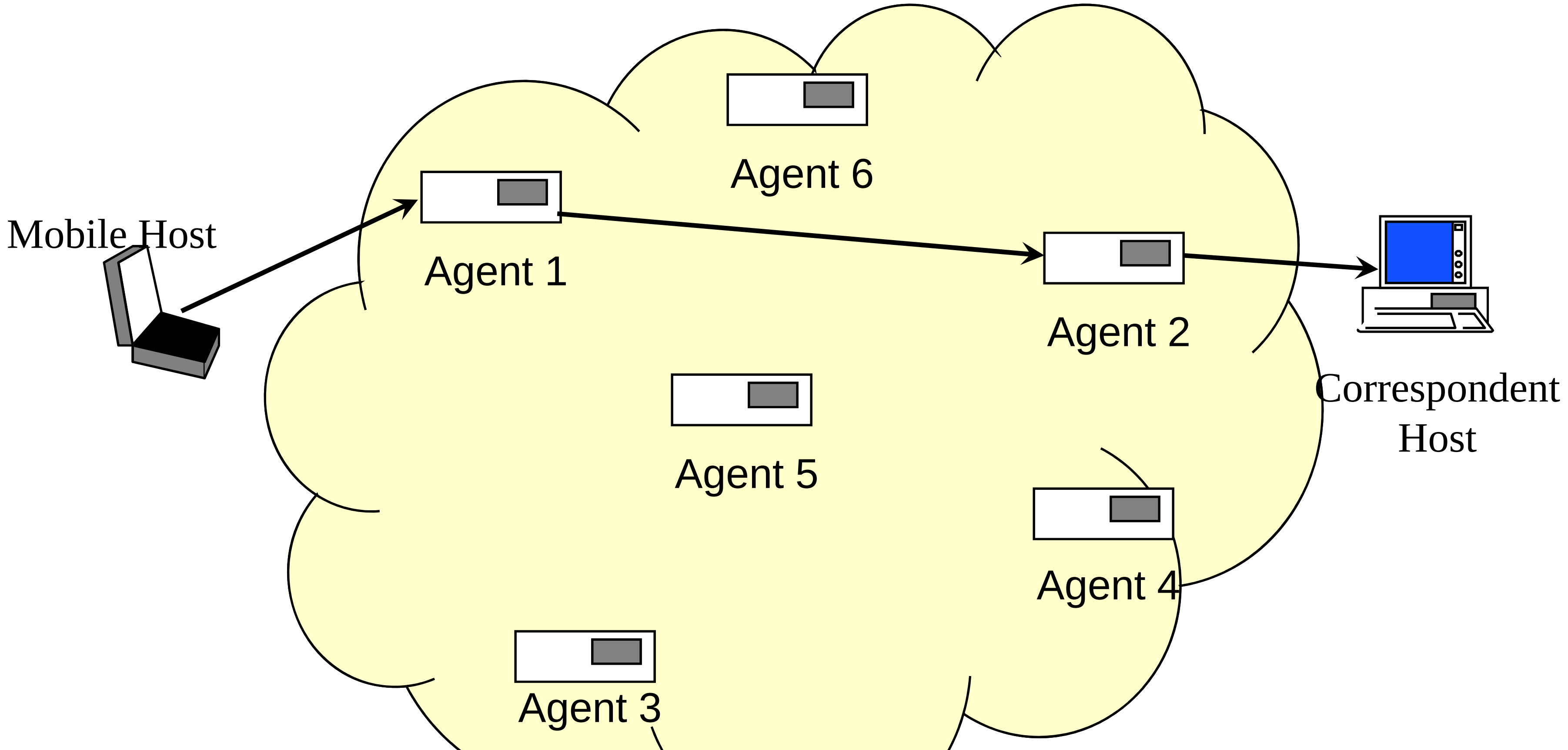
R1: path(@S,D,P,C) ← **closestAgent(@NI,S),** link(@S,D,C), P:=(S,D).
R2: path(@S,D,P,C) ← link(@S,Z,C_1), path(@Z,D,P_2,C_2), C:=C_1+C_2, P:=(S, P_2)
R3: bestPathCost(@S,D,min<C>) ← path(@S,D,Z,C).
R4: bestPath(@S,D,P,C) ← bestPathCost(@S,D,C), path(@S,D,P,C).
Query: bestPath(@S,D,P,C)

QoS-aware routing: low latency



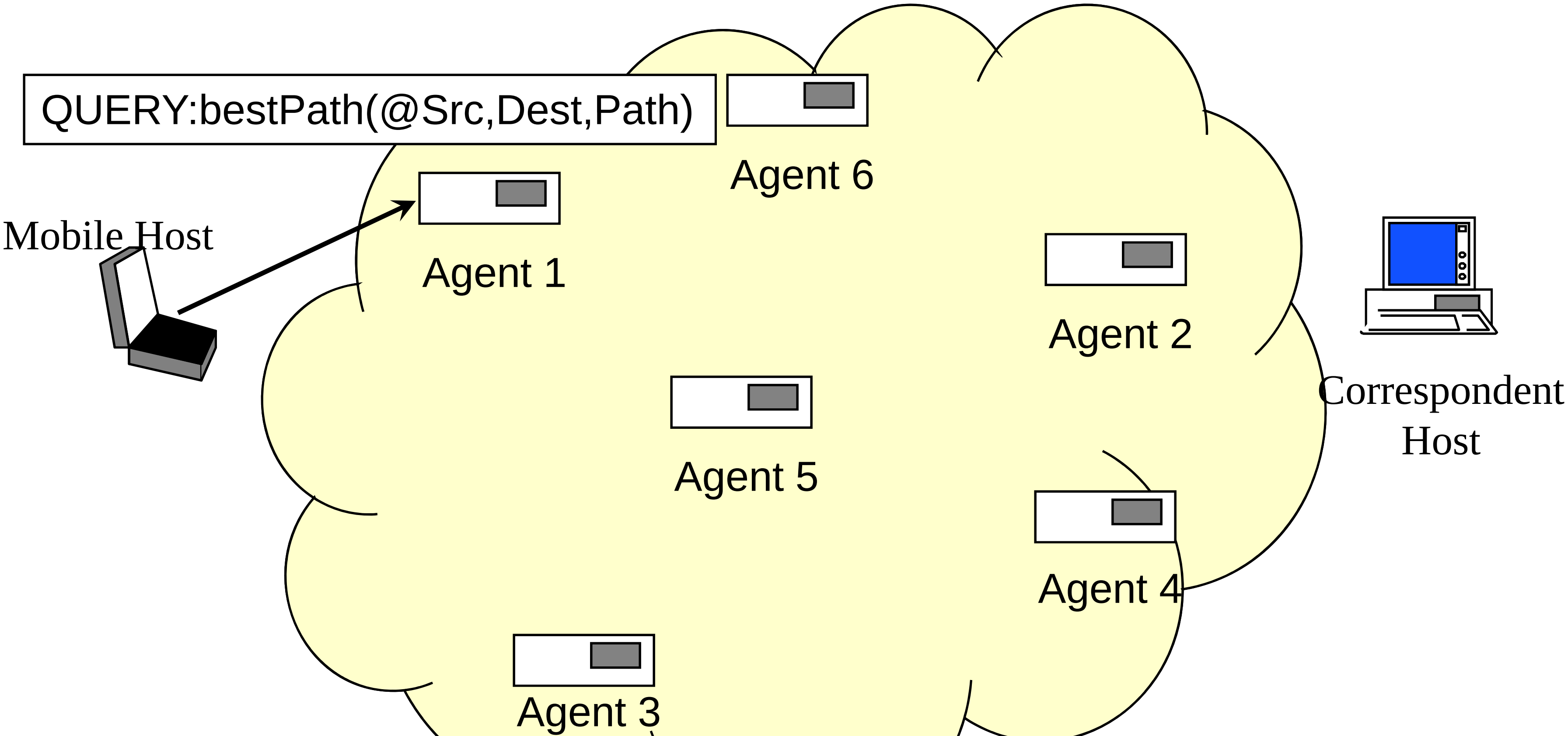
R1: path(@S,D,P,C) ← **closestAgent(@NI,S),** link(@S,D,C), P:=(S,D).
R2: path(@S,D,P,C) ← link(@S,Z,C_1), path(@Z,D,P_2,C_2), C:=C_1+C_2, P:=(S, P_2)
R3: bestPathCost(@S,D,min<C>) ← path(@S,D,Z,C).
R4: bestPath(@S,D,P,C) ← bestPathCost(@S,D,C), path(@S,D,P,C).
Query: bestPath(@S,D,P,C)

QoS-aware routing: low latency



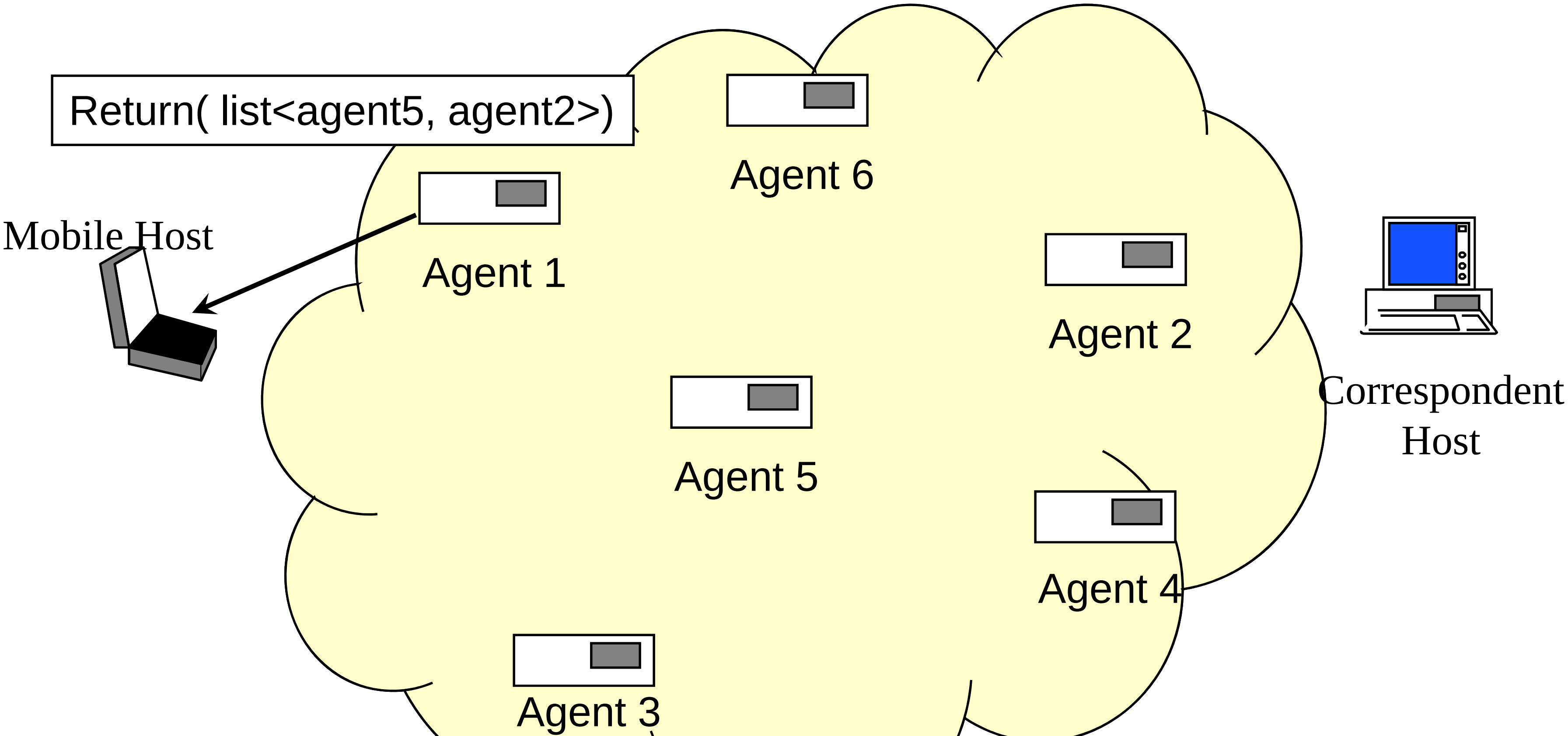
R1: path(@S,D,P,C) ← **closestAgent(@NI,S),** link(@S,D,C), P:=(S,D).
R2: path(@S,D,P,C) ← link(@S,Z,C_1), path(@Z,D,P_2,C_2), C:=C_1+C_2, P:=(S, P_2)
R3: bestPathCost(@S,D,min<C>) ← path(@S,D,Z,C).
R4: bestPath(@S,D,P,C) ← bestPathCost(@S,D,C), path(@S,D,P,C).
Query: bestPath(@S,D,P,C)

QoS-aware routing: high b/w



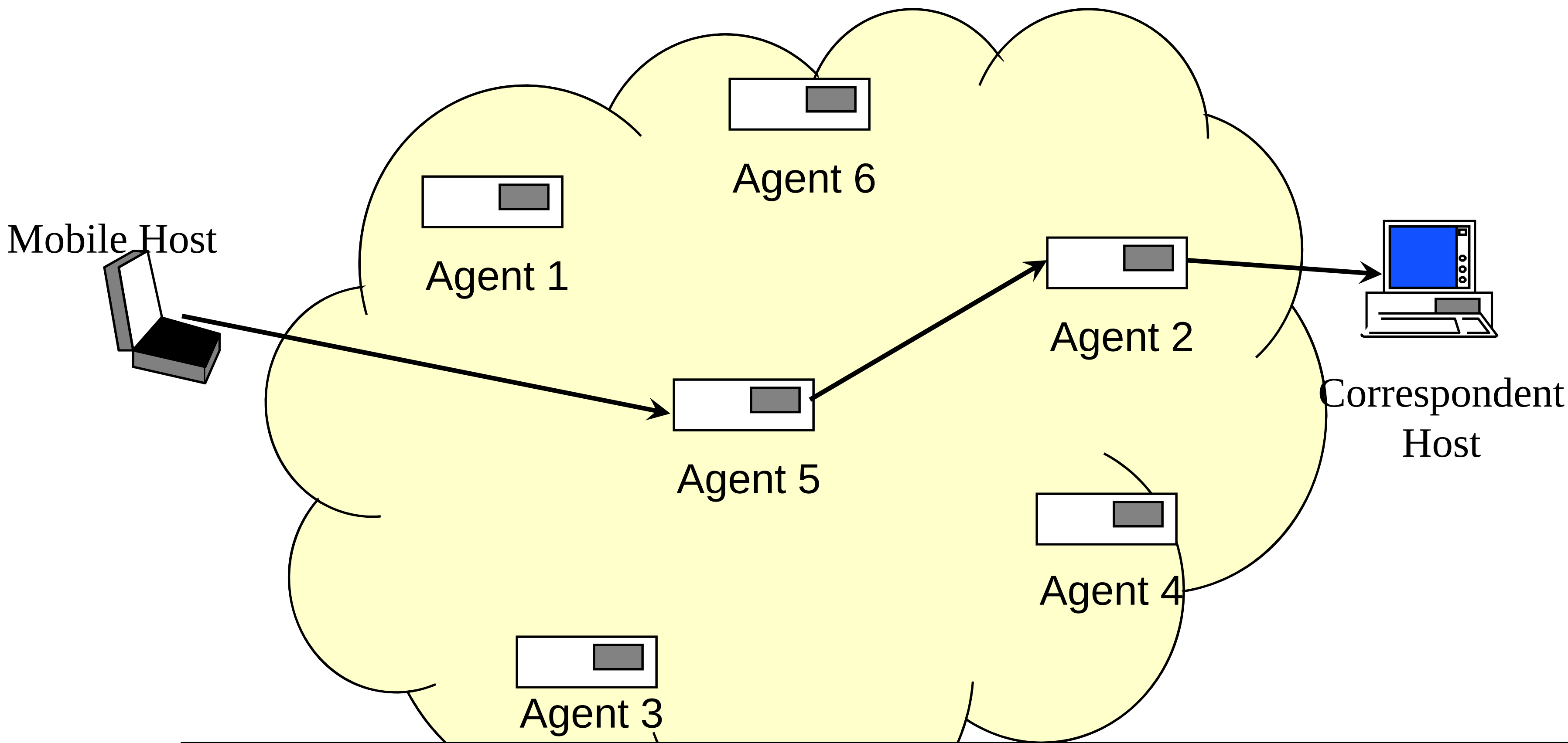
R1: path(@S,D,P,C) ← closestAgent(@NI, S), link(@S,D,C), P:=(S,D).
R2: path(@S,D,P,C) ← link(@S,Z,C_1), path(@Z,D,P_2,C_2), **C:=min(C_1,C_2)**, P:=(S, P_2)
R3: bestPathCost(@S,D,**max**<C>) ← path(@S,D,Z,C).
R4: bestPath(@S,D,P,C) ← bestPathCost(@S,D,C), path(@S,D,P,C).
Query: bestPath(@S,D,P,C)

QoS-aware routing: high b/w



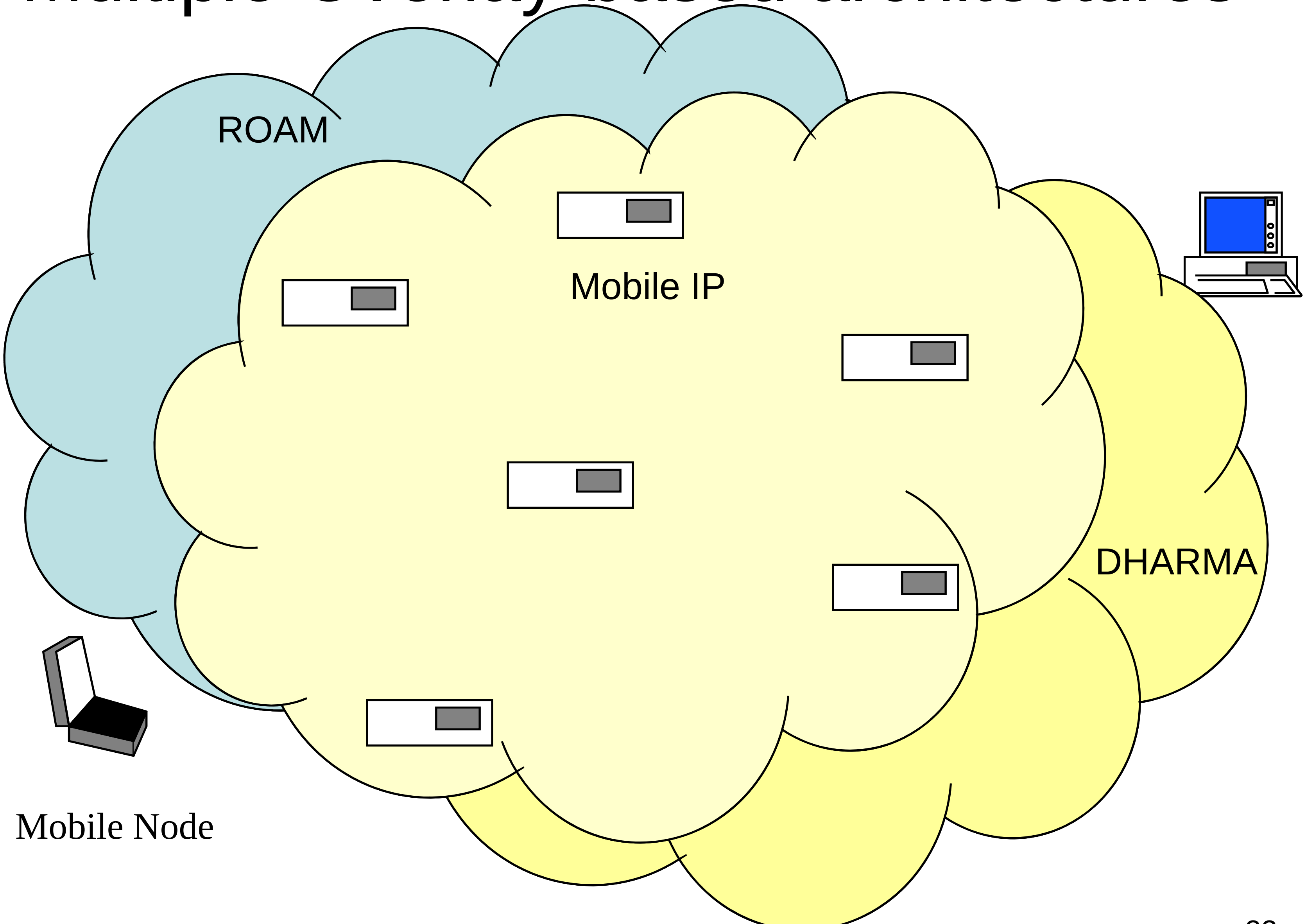
R1: path(@S,D,P,C) ← closestAgent(@NI, S), link(@S,D,C), P:=(S,D).
R2: path(@S,D,P,C) ← link(@S,Z,C_1), path(@Z,D,P_2,C_2), **C:=min(C_1,C_2)**, P:=(S, P_2)
R3: bestPathCost(@S,D,**max**<C>) ← path(@S,D,Z,C).
R4: bestPath(@S,D,P,C) ← bestPathCost(@S,D,C), path(@S,D,P,C).
Query: bestPath(@S,D,P,C)

QoS-aware routing: high b/w



R1: path(@S,D,P,C) ← closestAgent(@NI, S), link(@S,D,C), P:=(S,D).
R2: path(@S,D,P,C) ← link(@S,Z,C_1), path(@Z,D,P_2,C_2), **C:=min(C_1,C_2)**, P:=(S, P_2)
R3: bestPathCost(@S,D,**max**<C>) ← path(@S,D,Z,C).
R4: bestPath(@S,D,P,C) ← bestPathCost(@S,D,C), path(@S,D,P,C).
Query: bestPath(@S,D,P,C)

Multiple Overlay-based architectures



Implementation

- Built on P2 declarative networking system
- Built two overlay networks for mobility.
  - ROAM, DHARMA
- Tested on Emulab with 200 nodes
  - Performance comparable to the native implementation

Conclusions & Future work

- Rethink new mobile network abstractions
  - Data centricity, untangle the implementation
- Declarative architecture for mobile applications is promising
  - Extensible, flexible
  - Concise, efficient
- Future/ongoing Work
  - Application-aware mobile hosts
    - Declarative cross-layer abstraction for network stack
    - Declarative mobile ad-hoc networks
    - Legacy support
  - Additional useful features for mobility:
    - Flexible naming, late-binding service discovery, network monitoring
  - PlanetLab mobility service