IIP for Wireless Communication

Kirti Keshav and Venkataram P
ECE Department
Indian Institute of Science
Bangalore, 560012 INDIA
080-293-2282
kirti@protocol.ece.iisc.ernet.in, pallapa@ece.iisc.ernet.in

Abstract—This paper proposes IIP(Intelligent Internet Protocol) which effectively addresses the issues involved in wireless communication at Network Layer such as support for mobility, frequent loss of connection due to fading, handoffs and security. We have chosen to use Intelligent Agent technology for our architecture of IIP because it provides desirable properties such as Speed-up and Efficiency, Robustness and Reliability, Scalability and Flexibility, Development and Re-usability. We make use of four Intelligent Agents namely **Principal Agent(PA), Agent for Address Resolution and Mobility Support(AARMS), Agent for Error Handling and Recovery(AEHR), Agent for Network Security(ANS)** and design each Agent based on BDI(Belief-Desire-Intention) Architecture. A variant of Blackboard architecture is used for inter-agent communication. Special care has been taken to make proposed IP compatible with existing Networks. The proposed work has been simulated and verified for its operational effectiveness.

1. Introduction

While designing the architecture for Intelligent IP, we had four important design objectives, to make our IP really intelligent enough to solve the problems in wireless communication efficiently. This paper presents the architecture of Intelligent IP based on following design objectives:

Design objectives

- Intelligent Agent based IP has to be backward compatible with existing networks, i.e., it must handle all the existing network traffic successfully.
- Intelligent IP has to be flexible; for example, various functionalities of Network layer such as security, support for mobility are to be evoked only when needed in order to avoid excessive processing.
- Better cooperation and communication between protocol layers are key requirements for improving Internet protocol performance in future. Hence, we have to design in such a fashion that later on TCP functionality can be added to form a complete IP stack, based on layer less architecture using Intelligent Agents.
- Rapid development of new protocols. In future new protocols would be implemented quickly by adding Agents with new functionalities as required by new protocols.

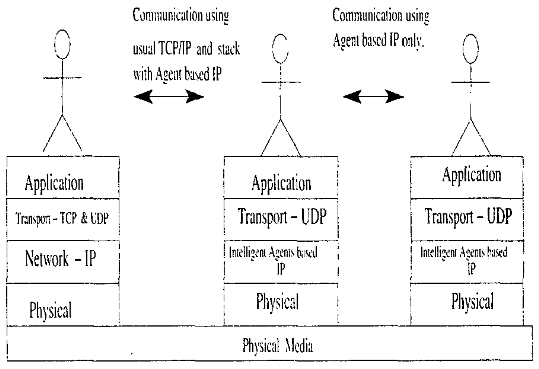
Communication using usual TCP/IP and stack with Agent based IP
Communication using Agent based IP only.
Application
Transport - TCP & UDP
Network - IP
Physical
Application
Transport - UDP
Intelligent Agents based IP
Physical
Application
Transport - UDP
Intelligent Agents based IP
Physical
Physical Media

Figure 1. A typical Communication Scenario

2. The proposed Intelligent IP for Wireless Communication

In our proposed IP, we make use of four Agents. Each individual Agent follows BDI Architecture as an internal model. Every Agent is able to sense the environment i.e., perceptions are same for all the Agents.

The IIP is intelligent in the sense that it learns from the environment dynamically and takes the best decision on the basis of present and past state information, among the various options available. A brief description of each of the four agents is as follows.

First Agent,which we name as **Principal Agent**(PA), takes care of normal IP functionality. It's possible actions include passing of arrived packet to upper layer protocol if destination of packet is host itself, forward packets to appropriate next hop if packet is meant for another host.

Second Agent, coined as **Agent for Address Resolution and Mobility Support**(AARMS), handles all addressing related issues of IP layer. It is responsible for maintenance of routing tables. If the environment is of Wireless type, then this Agent makes sure that handoffs and frequent temporary loss of connection in wireless link doesn't affect the upper layer connections. It can use Mobile IP for macro mobility (slow handoffs) and either cellular IP or Hierarchical Mobile IP in case of micro mobility(fast handoffs) to avoid excess signaling overhead. If other Agents need any information regarding IP addresses, they send request to this Agent. Packets corresponding to various Routing protocols such as RIP, OSPF, BGP are also to be handled by this Agent for maintain-

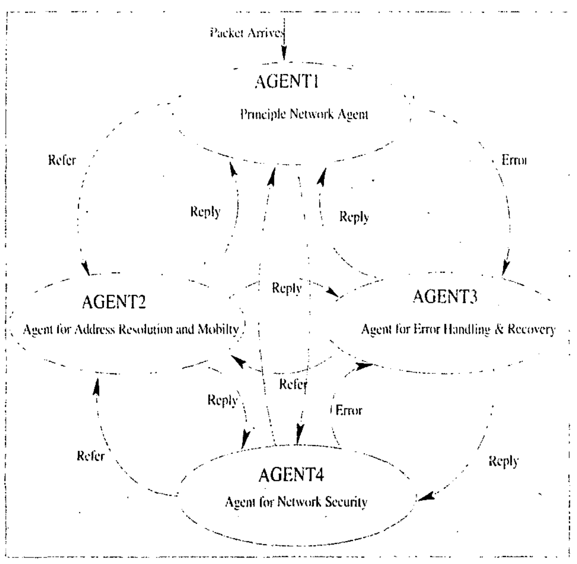

Figure 2. Intelligent IP for Wireless Communication

ing routing tables. In future routing protocol corresponding to Ad-Hoc wireless network are also to be handled by this Agent.

Third Agent, termed as **Agent for Error Handling and Recovery**(AEHR), is responsible for handling of various error situations. It handles all ICMP traffic and takes appropriate actions in response to it. It also helps **AARMS**, in maintaining routing information. For example, when ICMP Redirect message comes, suggesting a better next hop address to destination, it gives this information to AARMS.

Fourth Agent, Agent for Network Security (ANS). In Wireless Communication, since data is transmitted in open medium, security of data is extremely necessary. This Agent mainly takes care of all the four aspects of Network security,i.e., Secrecy, Authentication, Non-repudiation and Integrity Control. Currently, we propose to make use of IPSec protocol. Depending upon the current beliefs and intentions, it decides regarding which of the two types of security algorithms, symmetric encryption algorithms (e.g. Data Encryption Standard- DES) and One-way hash functions (e.g., Message Digest MD5 and Secured Hash Algorithm SHA-1), as recommended by IPSec protocol, is to be used. IPSec also provides ways to secure tunnels against false data origins, and encrypt traffic against unwanted network passive or active intruders from listening or modifying actions. Authentication, which is used for granting user access to corporate network and Encryption which is used to transfer user critical information through unreliable medium of the IP network are the most important aspects which are to be taken care of.

In addition, Principal Agent also acts as the facilitator that manages interaction between Agents for inter-agent communication.

3. Use of BDI Architecture: within each Agent

BDI architectures [1] are practical reasoning architectures, in which the process of deciding what to do resembles the kind of practical reasoning that we appear to use in our everyday lives. The basic components of a BDI architecture are data structures representing the beliefs, desires, and intentions of the agent, and functions that represent its deliberation (deciding what intentions to have – i.e., deciding what to do) and means-ends reasoning (deciding how to do it). Intentions play a central role in the BDI model: they provide stability for decision making, and act to focus the agent's practical reasoning.

We choose BDI Architecture as an internal architecture for each agent mainly for two reasons:

- It is intuitive - we all recognize the process of deciding what to do and then how to do it, and we all have an informal understanding of the notions of belief, desire, and intention.
- It gives us a clear functional decomposition, which indicates what sorts of subsystems might be required to build an Agent.

Decision making process of each Agent

If S is an arbitrary set, then

$$\wp(S)$$

is the power-set of S. The state of a BDI Agent at any given moment is, a triple (B,D,I), where

$$B \subset Bel, D \subset Des, andI \subset Int$$

An Agent's belief revision function is a mapping

$$brf : \wp(Bel) \times P \rightarrow \wp(Bel)$$

which on the basis of the current percept and current beliefs determines a new set of beliefs. The option generation function, *options*, maps a set of beliefs and a set of intentions to a set of desires.

$$options : \wp(Bel) \times \wp(Int) \rightarrow \wp(Des)$$

The purpose of *options* function is basically to perform means-ends reasoning with additional constraints of being *consistent*.

A BDI Agent's deliberation process (deciding what to do) is represented in the *filter* function,

$$filter : \wp(Bel) \times \wp(Des) \times \wp(Int) \rightarrow \wp(Int)$$

which updates the Agent's intentions on the basis of its previously held intentions, current beliefs and desires. *filter* should satisfy the following constraint:

$$\forall B \in \wp(Bel), \forall D \in \wp(Des), \forall I \in \wp(Int)$$

$$filter(B, D, I) \subset I \cup D.$$

In other words, current intentions are either previously held intentions or newly adopted options.

The *execute* function will then return executable intentions which correspond to directly executable actions:

$$execute : \wp(Int) \rightarrow A$$

The Agent decision function, *action* of a BDI Agent is then a function

$$actions : P \rightarrow A$$

and is defined by the following pseudo-code.

```
function action(p: P):A
begin
B := brf(B,p)
D := options(B,I)
I := filter(B,D,I)
return execute(I)
end function action
```

Some Beliefs of IIP's Agents

All Agents share set of common beliefs. Here is a small set of common beliefs which are useful at network layer.

- Whether communication is over wireless or wire-line link can be known by the IP addresses of communicating hosts.
- Loss rate
- Rate of environment change
- Security required or not
- Type of Application-Multimedia or Data Centric
- System related facts
  - Buffer space available
  - Processing power available

4. Method of Inter Agent Communication

In proposed architecture, communication between Agents is achieved using a Blackboard consisting of four regions, which are divided among four Agents. Each Agent has its own region, where it can write. But every other Agent has permission to read in memory areas of other agents. Due to this mechanism, every Agent knows what other Agents has to say. Synchronization among Agents is achieved by making use of global flags. Above scheme has been implemented using shared memory concept. Each of the Agent has been implemented using different process. A set of four memory areas are created by Principal Agent. Then these memory areas are attached by other Agents to their process address space.

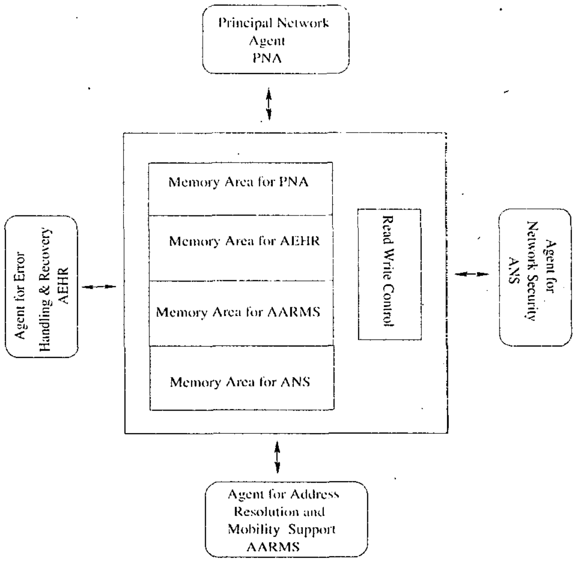
Principal Network Agent PNA
Memory Area for PNA
Memory Area for AEHR
Memory Area for AARMS
Memory Area for ANS
Read Write Control
Agent for Error Handling & Recovery AEHR
Agent for Network Security ANS
Agent for Address Resolution and Mobility Support AARMS

Figure 3. Inter Agent Communication

5. Implementation of IIP

Proposed Intelligent IP is implemented at Application layer in Linux Machine. Linux kernel implements a generic-purpose protocol, called PF_PACKET, which allows us to create a socket that receives packets directly from the network card driver. Hence, any other protocols handling is skipped, and any packets can be received. Using this PF_PACKET Socket facility, we are able to bypass the usual TCP/IP stack and receive the Ethernet frames directly at the Application layer. Since presently we are concentrating on IP layer only, at Transport layer we are using UDP.

Advantages of implementing IIP at Application layer

- PF_Packet Socket type through which we are able to receive packets directly at Application layer is a standard socket type, which is available on all Unix machines, so our implementation is portable on all Unix machines.
- At Application layer, we have access to facilities of modern high level language constructs. We are using facilities such as shared memory and processes. Each of our Agent is implemented as a process and to communicate among themselves, they use shared memory.

Now we present some case studies, to give an idea of how IIP reacts intelligently to different environment scenarios at network layer.

Case Studies

Each Agent in the system, gets the common perception from the environment. But since intentions of different Agents differ, each Agent responds independently to an event and updates itself with new options and intentions for future de-

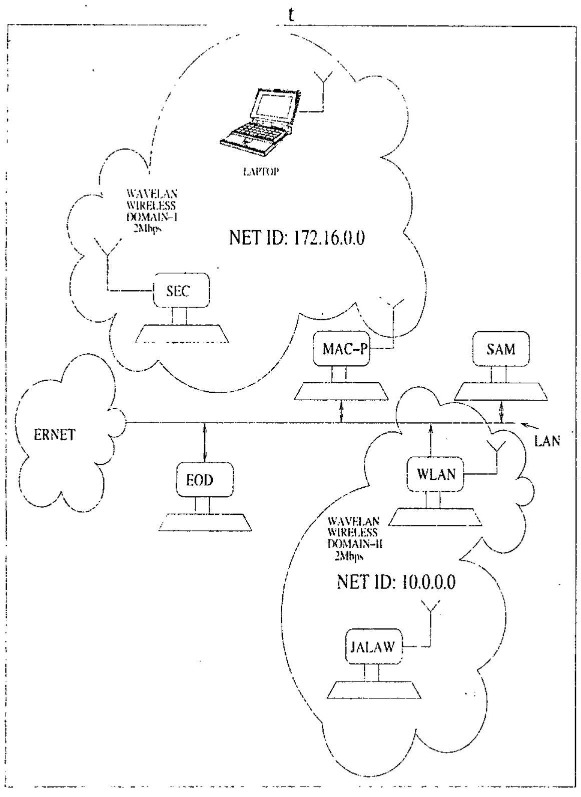

Figure 4. Wireless Network setup

cision. To make our implementation compatible to existing TCP/IP structure, we support [6], [7] and RFC791(for IP).

Case 1: Usual IP traffic is being received by our Intelligent IP and buffer overflow occurs in receiver's memory. Then AGENT3 sense this perception from environment, and sends ICMP_SOURCE_QUENCH message to sender.
Case 2: While receiving data if Agent2 finds that IP is not able to pass on the packets further, it sends back ICMP_UNREACH type message.
Case 3: If IP sends data and some intermediate router finds that better route is available then router sends an ICMP_REDIRECT message containing the address of better router. So, Agent2 will get this information from Agent3 and next packet onward AGENT1 will send packets with destination address mentioned in redirect message.
Case 4: If some intermediate hop finds that something is wrong with the header field of IP packet than it sends the source an ICMP_PARAMPROB message. On receiving of this message AGENT3 tries to locate the error and rectify it.
Case 5: If mobile node moves to a new network, AGENT2, which is responsible for address related issues , senses this event and initiate procedures to acquire care-of-address, so that mobile node can receive packets from Internet hosts without losing existing connections. After acquiring new care-of-address Agent2 intimates AGENT1 that it has received new care-of-address and it should now receive tunneled packets with new care-of-address.
Case 6: If a mobile node is about to make handoffs, then AGENT3 takes action by stopping the data transfer for a brief interval and start buffering the data being transferred. After hand-off has been completed, normal data transfer is resumed.
Case 7: Similarly if Agents sense from the environment that link quality is poor due to fading effects, then system starts taking precautionary measures by reducing the data transfer rate and buffering the data being transmitted, so that even if some packets are lost, they can be retransmitted locally only, instead of from the original source.

6. Wireless Network Setup & Experiments

We used the Wireless Network Setup as shown in **Figure 4**. Experiments were perfomed using the machines namely EOD, WLAN and JALAW. While WLAN is connected to network with both wireless and wireline link, JALAW is having only wireless link with network.

Processing Time

As a benchmarking test, an ICMP ECHO REQUEST coming from the EOD is received by WLAN and in response,WLAN generates an ICMP ECHO REPLY and send it back. The test was made by varying the size of payload on ECHO REQUEST message. When each message arrives or departs, the time was noted down with micro-seconds resolution at WLAN. Following graph depicts variation of average processing time with variation in payload size of arriving packets.

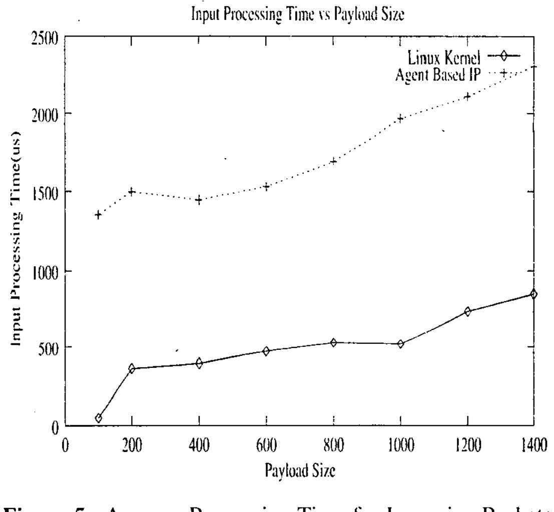

Figure 5. Average Processing Time for Incoming Packets

We can see that our Intelligent Agents based IP implementation's overall processing time for incoming packets is slightly more than that of Linux kernel's protocol stack implementation. Following reasons can be attributed to this phenomenon.

• We have implemented our Intelligent Agent Based IP at user level in Linux. So when a packet arrives at the Ethernet interface, kernel always gets the priority to handle packets over any other user process[10]. Because of this, even though our Agent Based IP receives packet directly from Ethernet interface, it gets packet after some delay.

• For transmitting the response packets also, since we are using PF_PACKET Socket facility, again user level process, which is obviously a slower method to transmit than the way kernel's TCP/IP stack does.

Round Trip Time

This test is to see average round trip time available with our implementation over a wireless link of bandwidth 2MBps. From EOD to JALAW on a 2 MBps Wireless link, we transmit ECHO_REQUEST message with varying payload. Both Server and Client are using our Intelligent Agent Based IP.

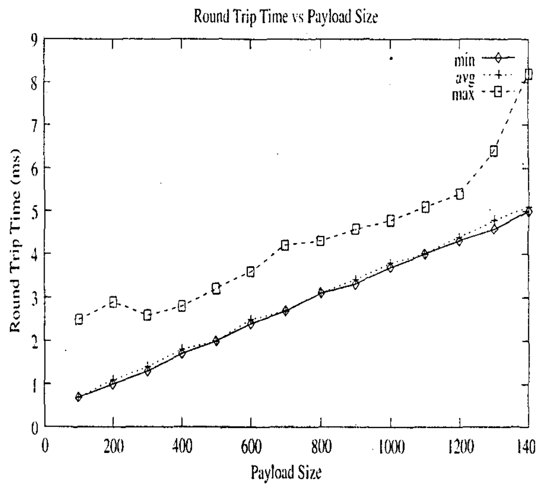

Figure 6. Round Trip Time for Packets with varying payload

The round-trip time estimation is a critical part of TCP, since the estimated round-trip time is used when determining a suitable retransmission time-out.

As one can note that we are able to achieve very less RTT (6-7 ms) with small packets, which shows that once TCP functionality is implemented, our Intelligent agent based IP would be able to handle interactive applications like Telnet, where response time upto 150ms is acceptable.

7. Results & Conclusion

We have deployed IIP in real-time TCP/IP environment where several FTP transactions, Multimedia stream transfers, etc., are taking place.To test our implementation, we performed various experiments and compared the parameters such as average processing time and round trip time.

From the results obtained, we can conclude that our **Intelligent IP for Wireless Communication** is compatible with existing TCP/IP stack and even though we have implemented our architecture at user level in Linux, thereby not having the advantage of faster handling of data as compared to Linux kernel, overall processing delays are comparable to existing TCP/IP implementation of Linux kernel. Also our Agent based architecture has added advantage of being flexible, scalable and reusable, as more Agents can be added easily as and when new requirements arise.

References

[1] Gerhard Weiss, "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence," The MIT Press, Massachusetts.

[2] http://www.computer.org/internet/v4n1/joy.htm

[3] M. E. Bratman, "Intentions, Plans, and Practical Reason," Harvard University. Press: Cambridge, MA, 1987.

[4] P. E. Agre and S. J. Rosenschein, "Computational Theories of Interaction and Agency," The MIT Press: Cambridge, MA, 1996.

[5] M. Wooldridge, "Agent-based software engineering," *IEE Transactions on Software Engineering*, 144(1):2637, February 1997.

[6] RFC793, T.Socolofsky, C.KaleA. "Transmission Control Protocol," September, 1981.

[7] RFC792, J.Postel, "Internet control Message Protocol," September, 1981.

[8] Glitho, R.H., Magedanz, T., "Applicability of Mobile Agents to Telecommunications," *IEEE Network*, Vol 16, Issue 3, May/Jun 2002.

[9] Stuart Russell and Peter Norvig, "Artificial Intelligence-A Modern Approach," Pearson Education, 2001.

[10] M.Beck, H.Bome, M.Dziadzka, U.Kunitz, R.Magnus, D Verworner, "Linux Kernel Internals," Addison-Wesley, 2000