Extended network tuples for the AA Protocol

Philip Budne
phil@Shiva.COM

Shiva Corporation
1 Cambridge Center
Cambridge, Ma 02142

ABSTRACT

This is a proposal for backwards compatible extensions to KIP style DDP in IP encapsulation routing (AA) packets for AppleTalk Phase 2 compatibility, using a familiar technique.

1. Goals

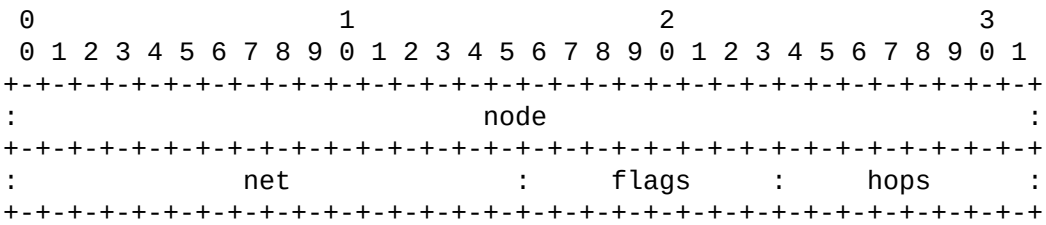
This proposal is being aired for comment, and the extensions presented here could be included (along with others) in a document for a new ‘‘IPTalk-2’’ standard.

The goals of this proposal are as follows;

- To allow ‘‘Kboxes’’ to exchange routes for extended networks using AA packets.
- To allow configuration of IP encapsulated routes for extended networks using `atalkatab`.
- To retain reasonable behavior with unmodified gateways and hosts.
- To incur the minimum reasonable implementation cost.
- To better specify the arouteTuple `flags` field.
- To allow configuration of EtherTalk Phase 2 ports.
- To allow more than 254 hosts of an IP (sub)net to appear as a single ‘‘IPTalk’’ network.
- To **NOT** define any new AA packets.

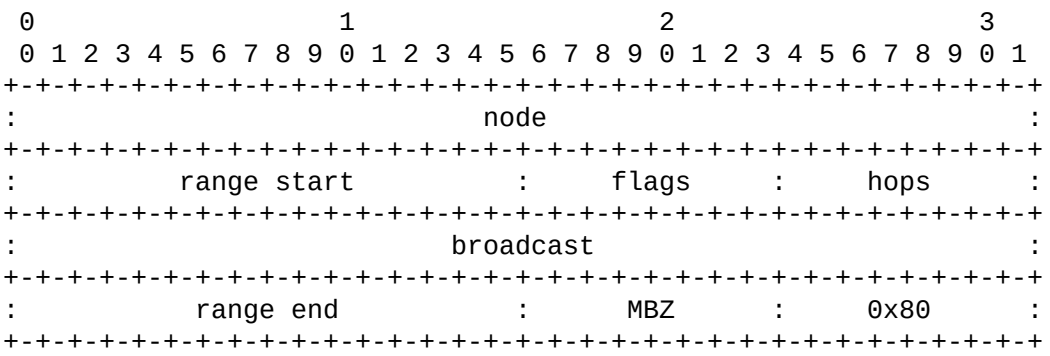
2. Representation of extended routes

Below is the routing tuple (called an arouteTuple) traditionally used by KIP and relatives to transfer routing information for ‘‘IPTalk’’.



Where `node` is an IP host or network address, and `net` is an AppleTalk DDP network number.

Extended networks are to be sent as two adjacent arouteTuples;



> *NOTE: An alternative would be to place the additional tuple FIRST which might make safe coding simpler — either way implementations MUST check that there are 2 tuples before checking for or obeying the special value of the* `hops` *field.*

Both arouteTuples for an extended network MUST fit within the 512 byte data region of an AA packet.

Existing implementations will discard the additional arouteTuple since it has an obscene hop count, and process only the first tuple.

2.1. `node` field

The `node` field contains the IP address to forward encapsulated packets for the indicated net range. For ‘‘Host’’ and ‘‘Net’’ routes, the final destination is calculated as follows;

```
dest_ip_addr  = tuple[0].node
              + ((ddp_dest_net - tuple[0].net) << 8)
              +   ddp_dest_node;
```

This enables IP networks with more than 254 host addresses to be represented with a single route (currently this can only be achieved using multiple ‘‘Net’’ lines and causes redundant IP broadcasts).

> *This means that aroute for the AppleTalk network range* `nn - mm` *will directly map onto the IP address range* `aa.bb.cc.0` *to* `aa.bb.(cc + (mm - nn)).255`*.*

Thus an `atalkatab` line of;

```
10-20   H   129.2.16.1   ZoneName
```

would map DDP addresses `0.10.0` to `0.20.255` onto the IP addresses `129.2.16.0` to `129.2.26.255`.

No modifications to CAP are required, as CAP always sends packets via a router (unless the DDP destination address matches the DDP source of the last packet received)!!

> *NOTE:*
>
> *IP Multicast [RFC1112] could be used for IPTalk broadcasts at some future time (to protect hosts not running CAP from NBP* `LkUp` *broadcasts). This proposal does not specify it’s use, since it has yet to become widely used or available in commercial offerings.*
>
> *Use of multicast would require that the NBP process on IPTalk nodes (ie;* `atis` *for CAP)*

register for reception of the appropriate Class-D address, and use of ‘‘Host’’ routes and `atalkrd`*, or a new special form of ‘‘Host’’ route which tells the router to use multicasts rather than broadcasts.*

Use of multiple (zone hashed) multicast address (as on Phase 2 EtherTalk and TokenTalk) or multi-network multicasts (ie; via DVMRP [RFC1075]) are not recommended.

The same methodology used above could be used to expand ‘‘Async’’ routes to allow more than 254 `async` program dialin users on a single host(!);

```
dest_udp_port = 0xAA00
              + ((ddp_dest_net - tuple[0].net) << 8)
              +   ddp_dest_node;
```

2.2. `range start` field

The `net` field of the first tuple represents the net range start of an extended network.

2.3. `flags` field

The only legal values for the `flags` field (for BOTH extended and non-extended tuples) are as follows.

`0x20` ‘‘Host’’
`0x22` ‘‘Async’’
`0x40` ‘‘Net’’ with zero low-order ones in broadcast
`0x41` ‘‘Net’’ with eight low-order ones in broadcast
`0x42` ‘‘Net’’ with sixteen low-order ones in broadcast
`0x43` ‘‘Net’’ with twenty-four low-order ones in broadcast
`0x80` ‘‘Kbox’’
`0x90` ‘‘Core Kbox’’

If an implementation does not recognize a value of the `flags` field it MUST ignore the route.

Gateways and AA host (ie; atalkad) implementations MUST NOT set the `aroute`AA flag `(0x08)` in any tuple sent in an AA packet.

2.4. `hops` field

Version zero implementations MUST discard all tuples with `hops` field values other than `0` to `15`.

Future backwards compatible extensions to version zero of this protocol might use this field in for advisory data (ie; route down notification).

2.5. `range end` field

The `net` field of the additional tuple represents the net range end of an extended network. Routers SHOULD interpret an extened tuple with a `range end` of zero as a non-extended network.

2.6. `version` field

The `hops` field of the additional tuple contains the extended tuple version, which MUST be `0x80` for the current version (version zero) of this protocol. Future versions of this protocol MAY use different values of this field, and version zero implementations MUST discard all tuples with `hops` values other than `0` through `15` and `0x80`.

NOTE: if any known implementations treat the `hops` *field as a* `signed char` *then a value less than* `0x80` *(ie;* `0x40` *or* `0x7F`*) would be a more prudent choice!*

2.7. `MBZ` field

The `flags` field of the additional arouteTuple MUST be sent as zero, and MUST be ignored on input.

Backwards compatible extensions to version zero of this protocol might use this field in for advisory data (ie; route time-to-live information).

2.8. `broadcast` field

If the `node` field of the additional arouteTuple is non-zero, it holds an IP destination address for DDP broadcasts (so that the the broadcast address for ‘‘Net’’ networks can be specified exactly). Implementations MAY ignore this field the route type is not ‘‘Net’’.

Here is an `atalkatab` line with an IP broadcast address specification for a subnet with mask `255.255.240.0` and broadcast address `129.2.31.255`;

start-end `N2B 129.2.16.0 129.2.31.255 zonelist...`

NOTE: The `2` *flag is included so that old routers can attempt delivery of packets for the first net in the range.*

3. `aaROUTE*` packets

All AA packets which normally carry arouteTuples MAY carry extended tuples (any of `aaROUTE`, `aaROUTEI`, `aaROUTEM`, or `aaROUTER`).

4. Phase 2 ZIP

4.1. `aaZONEQ` packets

`aaZONEQ` continues to be used for ZIP `Query, Reply,` and now, `Extended Reply`.

4.2. Phase 2 ZIP packets for IPTalk ports

The only ZIP packet sent by CAP (as of 6.0pl62) is `GetZoneList`. Although a Macintosh IPTalk ‘‘adev’’ (now possible with MacTCP 1.1) might find `GetLocalZones` and/or `GetNetInfo` useful.

5. NBP processing

Routers MUST process `FwdReq` packets for any network it advertises as extended.

`LkUp` packets MUST always be sent for nets with ‘‘Host’’ and ‘‘Net’’ routes (since the router which interprets the ‘‘Host’’ and ‘‘Net’’ is the last AppleTalk router to process the packet).

Processing `FwdReq` packets for IPTalk (‘‘Host’’ and ‘‘Net’’) networks entails only changing the type to `LkUp`, and sending the packet as an IP broadcast; neither the CAP libraries nor `atis` check the values of the DDP destination net and node, so there is no need to change the destination network to zero!

6. `atalkatab` syntax for extended networks

Here is an `atalkatab` line for a Phase 2 EtherTalk network connected to ‘‘Kbox’’ at *ipaddr*

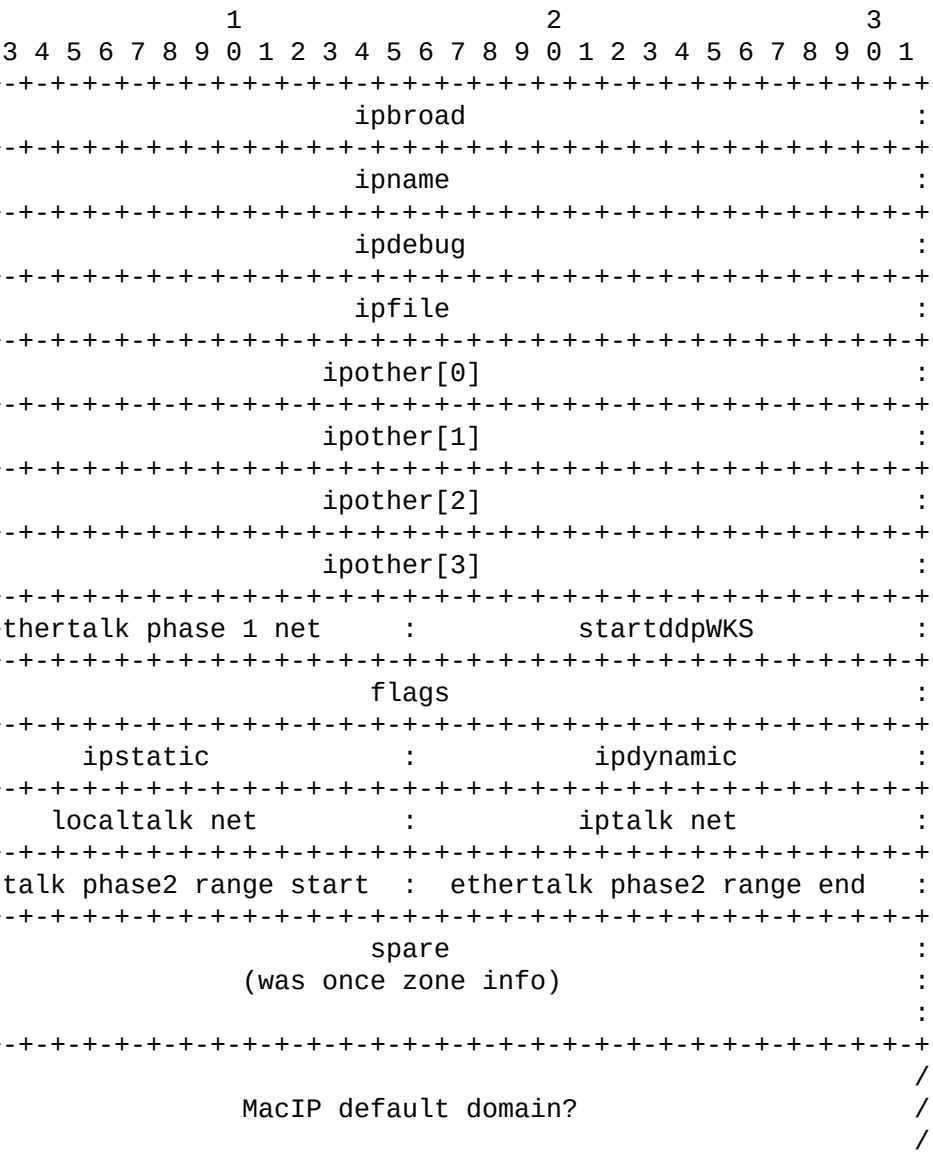
start-end `E` *ipaddr zonelist...*

The `B` flag is a modifier, and is not valid alone. It can be used with any type of line (but is most useful on `H` lines). Note that the broadcast address is also specified by the `2` for the benefit of unmodified routers.

Zone lists are constructed from whitespace separated tokens which may use underscore (ASCII 95) to represent spaces, and doubled underscore for underscore characters. If the zone name starts with an acute accent (ASCII 39) or quotation mark (ASCII 34) all following characters up to (but not including) a matching delimiter character are to be taken literally. Zone names may not span lines. If the line ends with a white space separated backslash character (ASCII 95), the next line will be taken as a continuation of the zone list. The continuation line may begin with whitespace. The K-STAR 9.0 `atalkatab` net ranges and zone lists on `E` lines.

7. Configuration info for EtherTalk Phase 2 ports

Configuration information for Kboxes is represented in `atalkatab` and sent in `aaCONF` packets. The format traditionally used by KIP and work-alikes is represented below. Configuration data in `atalkatab` is free-form and MAY not follow the following format;



The `ipname` field SHOULD be the address of a Domain Name Server [RFC1034,RFC1035] host, or MAY be the address of an IEN-116 Nameserver [IEN-116]. `ipname` SHOULD be sent to ‘‘MacIP’’ clients via IPGP (IPGATEWAY protocol) ATP packets.

The `ipfile` and four `ipother` fields MAY be used for implementation specific router configuration information, and SHOULD NOT be sent to ‘‘MacIP’’ clients via IPGP (see [MacIP-02]).

`Ethertalk phase2 range start` and `ethertalk phase2 range end` are extensions. The `0x08` bit in the `flags` field MUST be set to signal that this info is valid EtherTalk configuration info, since this area was once used for zone information (obsoleted in the 09/87 release of KIP!), and many `atalkatab` files still have zone strings there. The K-STAR 9.0 `atalkatab` parser already contains this extension.

The `startddpWKS` field is the base UDP port for ‘‘well known’’ DDP sockets. If this value is zero, a default value of 768 SHOULD be used. Implementations MAY use 200 as a default [RFC1060], and MUST allow selection of default value (minimally between 200 and 768).

NOTE: What about extended IPTalk net end? MacIP default domain name?

8. References

[IEN-116]
Postel, J. Name Server. 1979 August.

[MacIP-02]
Veizades, J. A Standard for the Transmission of Internet Packets Over AppleTalk Networks. draft document of December 1991. ftp.apple.com:pub/apple-ip/MacIP-02.txt

[RFC1034]
Mockapetris, P.V. Domain names - concepts and facilities. 1987 November.

[RFC1035]
Mockapetris, P.V. Domain names - implementation and specification. 1987 November

[RFC1060]
Reynolds, J.K.; Postel, J.B. Assigned numbers. 1990 March.

[RFC1075]
Waitzman, D.; Partridge, C.; Deering, S.E. Distance Vector Multicast Routing Protocol. 1988 November.

[RFC1112]
Deering, S.E. Host extensions for IP multicasting. 1989 August.

[RFC1243]
Waldbusser, S.,ed. Appletalk Management Information Base. 1991 July.