# Routing and Level 2 Addressing in a Hardware Accelerator for Network Applications

David Antoš*†
†CESNET, z. s. p. o.,
Zikova 4, Praha 160 00, Czech Republic
Email: {antos,rehak}@fi.muni.cz

Vojtěch Řehák*
*Faculty of Informatics,
Masaryk University Brno,
Botanická 68a, Brno 602 00, Czech Republic

***Abstract*— Personal computers are known to be highly usable as internet routers. To overcome their throughput limitations, a hardware accelerator can be employed. For the purposes of packet classification in the accelerator, we investigate a way to combine routing, level 2 addressing, and packet filtering into a single lookup structure.**

**This paper describes the first part of the process: a method to combine routing and level 2 addressing to a single lookup. A formal model of routing and level 2 addressing is presented and used to prove correctness of the method and its equivalence to operating system behaviour.**

## I. Introduction

Personal computers have shown that they are suitable as mid-sized routing platform [1]. Their throughput is comparable even with middle-class commercial routers. However, main limitation of their performance is the throughput of system bus. This bottleneck can be mitigated if packet switching is (at least partially) offloaded to an acceleration card, so that significant part of the network traffic handled by the accelerator only, avoiding buses inside the PC.

Operating systems perform packet classification in several stages: routing table lookup, packet filtering (usually spread into several stages at various places of the IP stack), and level 3 to level 2 address translation (neighbour cache or ARP table). The tables are usually maintained separately. Even open-source BSD clones that kept routing and ARP tables in a single structure tend to separate them [2].

*Packet classification*, i.e., deciding how to handle a packet, is the central part of the hardware routing accelerator. Because hardware resources limitations, we have designed a classification engine performing the classification in a single lookup operation. We have to combine routing tables, ARP tables, and packet filtering into a single structure. The first step to that goal is to combine routing and ARP tables. In this paper, we present a formalism to denote routing and ARP (Section III), a method to combine routing and ARP to a single lookup structure suitable for the accelerator and a method to obtain concise representation of the structure (Section IV).

Due to space limitations, full proofs of theorems presented in this paper can be found in [3].

## II. Related Work

Basic principles of sending packets over a subnetted network have been stated in [4]. Many methods to store routing tables have been investigated, we refer to the overview [5] for a detailed survey.

Information on processing routing and ARP in hardware boxes has been published in academic products only. Washington University Gigabit Switch [6] maintains the tables separately, similarly to majority of operating systems.

The formalism we developed is inspired by Frantzen's work on packet filters [7].

## III. Routing and ARP

This section describes how routing and ARP resolution are employed in the operating system and how they cooperate.

### A. Routing table

*Definition 1 (IP, routing table):* Let $IP$ be the set of IP addresses. Let $Interfaces$ be a finite set of network interfaces. Routing table is a function of IP addresses returning the interface and next hop IP address where to send the packet

$$R\colon IP \to Interfaces \times IP.$$

The way of address allocation reduces the number of prefixes. Network interfaces are connected to a network. The network shares the most significant bits of the address, the common part is called a *network prefix* and its length is a *network mask*. Prefixes may have arbitrary lengths as standardised by Classless Inter-Domain Routing (CIDR) [8].

To handle a packet, a router checks its destination address and finds the *longest matching prefix* in its routing table [4]. Routing table records are of two kinds, direct and indirect.

*Definition 2: Direct route* is a route leading to the network the router is directly connected to (a *local network* shares the same network prefix). Its next hop address is equal to its final destination. *Indirect route* is a route to the network that is accessed via an intermediate router, a gateway.

Although we have described routing tables as longest matching prefix structures, we define it formally as first match lists.

*Definition 3 (Routing table syntax):* A *routing table* $R$ is a sorted list of $\mathrm{Size}(R)$ rules. *Route* $R_i$ is the $i$-th rule for $1 \leq i \leq \mathrm{Size}(R)$ and $\mathrm{Len}(R_i)$ is the length of the prefix in that rule.

Let $[R_i]$ (which is a subset of $IP$) stand for the packets that match the $i$-th rule. If the routing table is represented with a trie structure, $[R_i]$ corresponds to the subtrie addressed by the prefix of $R_i$. $\mathrm{Len}(R_i)$ is then the length of the prefix. The

prefix is said to have *full length* if it contains a complete IP address (i.e., $\mathrm{Len}(R_i) = 32$ for IPv4 and 128 for IPv6).

Route $R_i$ is called *default* if $\mathrm{Len}(R_i) = 0$ (then $[R_i] = IP$). On the "output side," $\mathrm{NH_{IP}}(R_i)$ is the IP address of the next hop and $\mathrm{NH_{Int}}(R_i)$ is the output interface of the $i$-th rule.

If a destination address is not found in the routing table, the router drops the packet and informs the sender of the packet with "no route to host" error message.

We require, without loss of generality, that the routing table satisfies several conditions.

1) (First match) The rules are sorted in non-increasing prefix lengths. Formally, for $1 \leq i < j \leq \mathrm{Size}(R)$, condition $\mathrm{Len}(R_i) \geq \mathrm{Len}(R_j)$ holds. Note that this ordering can be obtained dumping the trie and sorting its records.
2) (Uniqueness) Prefixes of the same lengths are disjoint, i.e., for all $i$, $j$ such that $1 \leq i < j \leq \mathrm{Size}(R)$ and $\mathrm{Len}(R_i) = \mathrm{Len}(R_j)$ we have $[R_i] \cap [R_j] = \emptyset$.

The ordering allows us to define the semantics meaningfully.

*Definition 4 (Result of routing):* To obtain the result of routing for a destination address $p \in IP$, we find $R(p) = R_i$ where $i$ is the smallest index satisfying $p \in [R_i]$. Due to the ordering, it corresponds to finding the longest matching prefix of the destination address $p$.

The next hop address $\mathrm{NH_{IP}}(R(p))$ is the address of the gateway for indirect routes. For direct routes, next hop equals to the final destination: $\mathrm{NH_{IP}}(R(p)) = p$.

We will show that first match and trie representations can be converted. A first-match list equivalent to a given trie structure can be obtained by means of traversing the trie and sorting the rules depending on prefix lengths. The routing table contains at most one outcome for a particular prefix, therefore the first match and uniqueness properties are satisfied. In the opposite way, we just insert prefixes in the list into a trie. This is possible as prefixes of equal lengths are disjoint and the best matching prefix in the trie for an address is the first matching rule from the list.

We will need an extra property to be satisfied by the routing table. This property is purely technical. It will be necessary to allow us to re-arrange the order of records when routing and ARP tables are combined in Section IV.

3) (Relationship of direct and indirect networks) Direct networks do not contain indirect networks as their subnets with the exception of full length entries. Formally, for each $j$ such that $1 \leq j \leq \mathrm{Size}(R)$ and $R_j$ is direct we require that if $R_i$ exists for an $i$ such that $1 \leq i < j$ and $[R_i] \cap [R_j] \neq \emptyset$ then $R_i$ is also direct or full-length.

This requirement does not have to be satisfied by real-world routing tables. The table can be easily converted to comply with it by expanding the direct prefix, adding at most a number of prefixes determined by the difference of lengths of the conflicting prefix pair. Moreover, a single Depth First Search over the trie representation is sufficient to check and convert the table [3].

### B. Link Layer Addressing

In order to reach the destination host, a packet must be moved over each *link* on the path. In order to do so, the IP address of the next hop must be translated to its link address to build the link layer frame. Addresses on the link layer (also called *MAC, hardware, or physical addresses*) have flat structure (as opposed to the hierarchy of IP addresses).

*Definition 5 (ARP table):* ARP table is a function

$$A\colon IP \rightarrow MAC$$

where *MAC* is the set of physical addresses.

*ARP table* $A$ is a list of records. We denote *ARP records* as $A_i$ for $1 \leq i \leq \mathrm{Size}(A)$. The ordering of rules is not important here as we require each set $[A_i]$ to contain just a single IP address. We require the records to be unique, i.e., for $1 \leq i < j \leq \mathrm{Size}(A)$ the condition $[A_i] \cap [A_j] = \emptyset$ holds. The result of the ARP lookup is the MAC address $\mathrm{NH_{MAC}}(A_i)$.

The *result of ARP lookup* for an address $p \in IP$ is the record $A(p) = A_i$ such that $p \in [A_i]$.

The ARP table does not have to cover the complete IP address space. If the corresponding MAC address is not found for the next hop, ARP protocol [9] is used to learn it.

## IV. Routing and ARP Combination

In this section, we describe a method to combine routing and level 2 addressing into a single lookup operation.

### A. Software Cooperation

An essential principle is used in the design: *If the hardware lookup cannot resolve a packet itself it can send it to software in the same way as an ordinary network adapter would.*

The accelerator behaves as an ordinary network adapter card from the operating system's point of view, only switching some traffic by itself. The task is to make behaviour of the accelerator equivalent to the behaviour of the operating system.

In the algorithms, this sending the packet to the operating system will be denoted by a "*SW*" action. Of course, packets passing through software are processed more slowly than packets switched by the hardware engine. It does not have to indicate a serious problem if the portion of the traffic processed by the operating system related to the total traffic is small enough.

### B. RA Table Definition

The structure combining routing and ARP is called a *routing-ARP (RA) table*:

$$RA\colon IP \rightarrow (Interfaces \times MAC) \cup \{SW\}$$

The RA table returns either the interface and MAC address of the next hop where the packet should be sent or it indicates that the packet must be processed by software.

*Definition 6 (RA table): Routing-ARP table* $RA$ is an ordered list of rules. We will use $RA_i$ for *routing-ARP rules* (for $1 \leq i \leq \mathrm{Size}(RA)$) and $[RA_i]$ for IP addresses affected by the rule. Analogically with routing syntax, $\mathrm{NH_{Int}}(RA_i)$

and $\mathrm{NH}_{\mathrm{MAC}}(RA_i)$ are the *interface* to reach the next hop and the *hardware address of the next hop*, respectively.

We define $\mathrm{Len}(RA_i)$, default and full-length records similarly as for the routing table.

To obtain a *result* for a destination address $p \in IP$, denoted $RA(p)$, we find $RA_i$ for the smallest $i$ such that $p \in [RA_i]$.

### C. Computing First-Match RA Tables

To create the routing-ARP table, we "inject" ARP entries inside the routing table. In case of locally connected network, we add all resolved ARP entries and finally we send the rest of the network to the operating system (in order to resolve ARP and/or emit an error). For indirect routes we insert the MAC address of the next hop into the table directly instead of its IP address if the MAC address is known. Otherwise the route must be processed by the operating system. The operating system runs the ARP protocol to learn the appropriate record. Changes in the ARP table and/or the routing table cause the RA table to recompute.

Algorithm in Fig. 1 is a pseudocode for combining routing table represented as a list (which is equivalent to traversing a trie in non-increasing prefix lengths) and ARP table (represented in an arbitrary manner, say a list) into the RA table represented as a list.

We have to show that the RA result is the same as ordinary software processing where lookups in routing table and resolving ARP are performed separately. Sending the packet to software is also a correct result—in that case the packet is processed by the original routing and ARP tables. We start with an observation that Fig. 1 performs "useful work," i.e., at least some traffic is accelerated. For each direct route, routing-ARP records are created for all destinations if the appropriate ARP record is known. If the ARP record of a gateway is known it is used to create a routing-ARP record for the indirect route.

*Lemma 1:* RA table produced by algorithm in Fig. 1 is total (i.e., it is defined for each destination address $p \in IP$).

*Proof:* The algorithm inserts the default RA rule. We recall that full versions of all proofs can be found in [3]. ∎

We have shown that the result of RA lookup is defined. In the following theorem, we show that the result is correct.

*Theorem 1 (Correctness of RA compilation):* For each destination address $p \in IP$, routing-ARP table contains a result that is either $SW$ or it is the same as applying routing and then ARP on the destination address, i.e., $\mathrm{NH}_{\mathrm{Int}}(RA(p)) = \mathrm{NH}_{\mathrm{Int}}(R(p))$ and $\mathrm{NH}_{\mathrm{MAC}}(RA(p)) = \mathrm{NH}_{\mathrm{MAC}}(A(\mathrm{NH}_{\mathrm{IP}}(R(p))))$.

*Proof:* By discussing origins of all RA rules. ∎

We have obtained a routing-ARP table that is behaviourally equivalent to the original routing and ARP tables. The RA table is first match but it can contain redundancies. It can be advantageous to have a trie representation of the table (i.e., an equivalent longest matching prefix representation) in order to obtain a concise representation.

If we are able to sort RA entries in a non-increasing way and ensure that prefixes of equal lengths are disjoint we would obtain a table that can be converted into a trie form. To reach this goal, let us study the ordering and relations of rules produced by the algorithm and potential redundancies in the table that have to be discarded.

```
RA = ∅
l = 1
for i = 1 to Size(R) do
   if R_i is direct
      then
         /* go through ARP records in this local subnet: */
         for each j such that 1 ≤ j ≤ Size(A)
                and [A_j] ⊆ [R_i] do
            [RA_l] = [A_j]
            RA_l = (NH_Int(R_i), NH_MAC(A_j))
            l = l + 1
         done
         /* the rest must get resolved in software */
         [RA_l] = [R_i]
         RA_l = SW
         l = l + 1
      else /* R_i is indirect */
         if exists k such that NH_IP(R_i) = [A_k]
            then
               [RA_l] = [R_i]
               RA_l = (NH_Int(R_i), NH_MAC(A_k))
               l = l + 1
            else
               [RA_l] = [R_i]
               RA_l = SW
               l = l + 1
         fi
   fi
done
if l = 1 ∨ [RA_{l-1}] ≠ IP /* ∨ requires lazy evaluation */
   [RA_l] = IP /* ensure that RA contains a default route */
   RA_l = SW
fi
```

Fig. 1. Combining routing and ARP into RA table

*Theorem 2 (RA compilation sort order):* Algorithm in Fig. 1 sorts the resulting RA table in non-increasing prefix lengths with exception of prefixes of full lengths, i.e., for $1 \leq k < l \leq \mathrm{Size}(RA)$ the following condition holds: $\mathrm{Len}(RA_k) \geq \mathrm{Len}(RA_l)$ or $RA_l$ has full length. Moreover, prefixes with equal lengths with exception of full-length prefixes are disjoint.

*Proof:* The algorithm preserves lengths of prefixes and/or expands them to full lengths. ∎

The full-length prefix for a particular destination address may be repeated across the table several times. In that case the first occurrence is of interest as all the following ones are unreachable and may be omitted. For formal description, see Lemma 6 of [3].

Full-length prefixes are mixed into prefixes of other lengths.

This algorithm is the same as the algorithm in Fig. 1 with the following modifications.

- The resulting RA table is kept in a trie structure. (Hence it is sorted on-the-fly in non-increasing way and all records are unique.)
- Let us understand assignments that create rules $RA_l$ in the following way:
  - Assignment to $[RA_l]$ is creating a path in the trie denoting the prefix of the $RA_l$ rule.
  - Assigning the output to the rule (e.g., $RA_l = (\mathrm{NH}_{\mathrm{Int}}(R_i), \mathrm{NH}_{\mathrm{MAC}}(A_j))$) means *assign the output only if it was empty previously*.

Fig. 2. Combining routing and ARP into RA table expressed as trie

In order to prepare the table to be converted into longest prefix semantics it would be more suitable to move the prefixes to the beginning of the table. To ensure that this does not change the meaning of the table, it is enough that the address space affected by the prefix is disjoint with all the preceding rules (i.e., rules of arbitrary lengths). We have to be interested in the *first* occurrence of the full-length prefix for an address only.

The following result ensures that the first occurrence of a full length entry in the RA table is disjoint with *all* its predecessors. From the operational point of view, the first occurrence of a full length entry for an address is *reachable* in the routing-ARP table.

*Theorem 3 (Sorting full-length RA records):* Let $RA_l$ be the first occurrence of full-length RA record for a given destination address, precisely, let $RA_l$ be a full-length RA record satisfying that for each full-length record $RA_m$ such that $[RA_l] = [RA_m]$ we have $l \leq m$. Then for each $k$ such that $1 \leq k < l$ condition $[RA_k] \cap [RA_l] = \emptyset$ holds.

*Proof:* By discussing all relationships between pairs of prefixes. This proof is very technical and we urge a curious reader to study it in [3]. The property "Relationship of direct and indirect networks" is necessary to prove the theorem and we refer to the technical report for an example how this property can be violated in real world tables. ■

### D. Longest Matching Prefix Representation of the RA Table

We have observed following properties of the routing-ARP table produced by the algorithm in Fig. 1:

1) Only the first occurrence of an address space is of interest, precisely if the algorithm creates an entry $RA_l$ and an entry $RA_k$ such that $[RA_k] = [RA_l]$ and $k < l$ has been created before, we do not have to store $RA_l$ into the RA table.
2) Full-length entries may be pushed to the beginning of the table thanks to Theorem 3, obtaining a table sorted in non-increasing prefix lengths.

Instead of post-processing the table, we may change the order of the rules immediately during the run of the algorithm with a small change of semantics of assignments. The final version of the algorithm is shown in Fig. 2. With those changes, the algorithm constructs an equivalent trie representation of the RA table.

## V. Conclusion

The paper brings two main contributions: stating a formal model of routing and ARP from the packet switching point of view and giving a method to combine routing and level 2 addressing into a single lookup operation. Special interest has been given to showing correctness of the approach with respect to rule reordering to obtain a concise representation of the RA structure that is suitable for further processing. Although various combinations of routing and ARP tables are used in practice (mainly in BSD operating systems), their behaviour has not been studied formally before.

The property on relationship of direct and indirect routes is a point of possible improvements. Although we have a simple method to convert a routing table to the required form, it would be interesting to incorporate the condition directly into the algorithm and to compare it to processing "in two stages."

Results described in this paper have been used as a base for a method to combine routing, ARP, and packet filtering into a single lookup operation. The idea behind adding the packet filter is based on Interval Decision Diagram (IDD) representation of the filter [10]. We combine the IDD representation with the RA table, distributing the filters into relevant portions of the address space. The resulting structure is employed in the hardware accelerator COMBO6 [1].


## Acknowledgement

This project has been supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201). The work of Vojtěch Řehák has been partially supported by the Academy of Sciences of the Czech Republic grant No. 1ET408050503.


## References


[1] J. Novotný, O. Fučík, and D. Antoš, "Project of IPv6 Router with FPGA Hardware Accelerator," in *Field-Programmable Logic and Applications, 13th International Conference FPL 2003*, P. Y. Cheung, G. A. Constantinides, and J. T. de Sousa, Eds., vol. 2778. Springer Verlag, September 2003, pp. 964–967.

[2] L. Rizzo, "New arp code snapshot for review," Apr 2004, Mailing list FreeBSD-current, http://lists.freebsd.org/pipermail/freebsd-current/2004-April/026380.html.

[3] D. Antoš, "Combining Routing and ARP to a Single Lookup Operation," CESNET, z. s. p. o., Tech. Rep. 4/2005, August 2005.

[4] J. C. Mogul and J. Postel, "RFC 950: Internet Standard Subnetting Procedure," Aug. 1985, updates RFC0792.

[5] D. Antoš, "Overview of Data Structures in IP Lookups," CESNET, Tech. Rep. 9/2002, September 2002.

[6] F. Kuhns, J. DeHart, A. Kantawala, R. Keller, J. Lockwood, P. Pappu, D. Richard, D. Taylor, J. Parwatikar, E. Spitznagel, J. Turner, and K. Wong, "Design of a high performance dynamically extensible router," in *Proceedings of DARPA Active Networks Conference and Exhibition*, San Francisco, CA, USA, May 2002.

[7] L. Frantzen, "Approaches for Analysing and Comparing Packet Filtering in Firewalls," Master's thesis, Technical University of Berlin, May 2003.

[8] V. Fuller, T. Li, J. Yu, and K. Varadhan, "RFC 1519: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy," Sept. 1993, obsoletes RFC1338.

[9] D. C. Plummer, "RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware," Nov. 1982.

[10] M. Christiansen and E. Fleury, "An Interval Decision Diagram Based Firewall," in *Proceedings of 3rd IEEE International Conference on Networking (ICN '04)*. Gosier, Guadeloupe, French Caribbean: University of Haute Alsace, Colmar, France, 2004, ISBN 0-86341-325-0.