EXHIBIT C

Network Working Group D. L. Mills
Request for Comments: 981 M/A-COM Linkabit
March 1986

An Experimental Multiple-Path Routing Algorithm

Status of This Memo

This RFC describes an experimental, multiple-path routing algorithm designed for a packet-radio broadcast channel and discusses the design and testing of a prototype implementation. It is presented as an example of a class of routing algorithms and data-base management techniques that may find wider application in the Internet community. Of particular interest may be the mechanisms to compute, select and rank a potentially large number of speculative routes with respect to the limited cumputational resources available. Discussion and suggestions for improvements are welcomed. Distribution of this memo is unlimited.

Abstract

This document introduces wiretap algorithms, which are a class of routing algorithms that compute quasi-optimum routes for stations sharing a broadcast channel, but with some stations hidden from others. The wiretapper observes the paths (source routes) used by other stations sending traffic on the channel and, using a heuristic set of factors and weights, constructs speculative paths for its own traffic. A prototype algorithm, called here the Wiretap Algorithm, has been designed for the AX.25 packet-radio channel. Its design is similar in many respects to the shortest-path-first (spf) algorithm used in the ARPANET and elsewhere, and is in fact a variation in the class of algorithms, including the Viterbi Algorithm, that construct optimum paths on a graph according to a distance computed as a weighted sum of factors assigned to the nodes and edges.

The Wiretap Algorithm differs from conventional algorithms in that it computes not only the primary route (a minimum-distance path), but also additional paths ordered by distance, which serve as alternate routes should the primary route fail. This feature is also useful for the discovery of new paths not previously observed on the channel.

Since the amateur AX.25 packet-radio channel is very active in the Washington, DC, area and carries a good deal of traffic under punishing conditions, it was considered a sufficiently heroic environment for a convincing demonstration of the prototype algorithm. It was implemented as part of an IP/TCP driver for the LSI-11 processor running the "fuzzball" operating system. The driver is connected via serial line to a 6809-based TAPR-1 processor running the WA8DED firmware, which controls the radio equipmnet in both

virtual-circuit and datagram modes. The prototype implementation provides primary and alternate routes, can route around congested areas and can change routes during a connection. This document describes the design, implementation and initial testing of the algorithm.

1. Introduction

This document describes the design, implementation and initial testing of the Wiretap Algorithm, a dynamic routing algorithm for the AX.25 packet-radio channel [4]. The AX.25 channel operates in CSMA contention mode at VHF frequencies using AFSK/FM modulation at 1200 bps. The AX.25 protocol itself is similar to X.25 link-layer protocol LAPB, but with an extended frame header consisting of a string of radio callsigns representing a path, usually selected by the operator, between two end stations, possibly via one or more intermediate packet repeaters or digipeaters. Most stations can operate simultaneously as intermediate systems digipeaters) and as end systems with respect to the ISO model.

Wiretap uses passive monitoring of frames transmitted on the channel in order to build a dynamic data base which can be used to determine optimum routes. The algorithm operates in real time and generates a set of paths ordered by increasing total distance, as determined by a shortest-path-first procedure similar to that used now in the ARPANET and planned for use in the new Internet gateway system [2]. The implementation provides optimum routes (with respect to the factors and weights selected) at initial-connection time for virtual circuits, as well as for each datagram transmission. This document is an initial status report and overview of the prototype implementation for the LSI-11 processor running the "fuzzball" operating system.

The principal advantage in the use of routing algorithms like Wiretap is that digipeater paths can be avoided when direct paths are available, with digipeaters used only when necessary and also to discover hidden stations. In the present exploratory stage of evolution, the scope of Wiretap has been intentionally restricted to passive monitoring. In a later stage the scope may be extended to include the use of active probes to discover hidden stations and the use of clustering techniques to manage the distribution of large quantities of routing information.

The AX.25 channel interface is the 6809-based TAPR-1 processor running the WA8DED firmware (version 1.0) and connected to the LSI-11 by a 4800-bps serial line. The WA8DED firmware produces as an option a monitor report for each received frame of a selected type,

including U, I and S frames. Wiretap processes each of these to extract routing information and (optionally) saves them in the system log file. Following is a typical report:

```
fm KS3Q to W4CQI via WB4JFI-5* WB4APR-6 ctl I11 pid F0
```

The originating station is KS3Q and the destination is W4CQI. The frame has been digipeated first by WB4JFI-5 and then WB4APR-6, is an I frame (sequence numbers follow the I indicator) and has protocol identifier F0 (hex). The asterisk "*" indicates the report was received from that station. If no asterisk appears, the report was received from the originator.

2. Design Principles

A path is a concatenation of directed links originating at one station, extending through one or more digipeaters and terminating at another station. Each link is characterized by a set of factors such as cost, delay or throughput that can be computed or estimated. Wiretap computes several intrinsic factors for each link and updates the routing data base, consisting of node and link tables. The weighted sum of these factors for each link is the distance of that link, while the sum of the distances for each link in the path is the distance of that path.

It is the intent of the Wiretap design that the distance of a link reflect the a-priori probability that a packet will successfully negotiate that link relative to the other choices possible at the sending node. Thus, the probability of a non-looping path is the product of the probabilities of its links. Following the technique of Viterbi [1], it is convenient to represent distance as a logarithmic transformation of probability, which then becomes a metric. However, in the following the underlying probabilities are not considered directly, since the distances are estimated on a heuristic basis.

Wiretap incorporates an algorithm which constructs a set of paths, ordered by distance, between given end stations according to the factors and weights contained in the routing data base. Such paths can be considered optimum routes between these stations with respect to the given assignment of factors and weights. In the prototype implementation one of the end stations must be the Wiretap station itself; however, in principle, the Wiretap station can generate routes for other stations subject to the applicability of the information in its data base.

Note that Wiretap in effect constructs minimum-distance paths in the

direction from the destination station to the Wiretap station and, based on that information, then computes the optimum reciprocal routes from the Wiretap station to the destination station. The expectation is that the destination station also runs its own routing algorithm, which then computes its own optimum reciprocal routes (i.e. the optimum direct routes from the Wiretap station). However, the routing data bases at the two stations may diverge due to congestion or hidden stations, so that the computed routes may not coincide.

In principle, Wiretap-computed routes can be fine-tuned using information provided not only by its directly communicating stations but others that may hear them as well. The most interesting scenario would be for all stations to exchange Wiretap information using a suitable distributed protocol, but this is at the moment beyond the scope of the prototype implementation. Nevertheless, suboptimum but useful paths can be obtained in the traditional and simple way with one station using a Wiretap-computed route and the other its reciprocal, as determined from the received frame header. Thus, Wiretap is compatible with existing channel procedures and protocols.

3. Implementation Overview

The prototype Wiretap implementation for the LSI-11 includes two routines, the wiretap routine, which extracts information from received monitor headers and builds the routing data base, and the routing routine, which calculates paths using the information in the data base. The data base consists of three tables, the channel table, node table and link table. The channel table includes an entry for each channel (virtual circuit) supported by the TAPR-1 processor running the WA8DED firmware, five in the present configuration. The structure and use of this table are only incidental to the algorithm and will not be discussed further.

The node table includes an entry for each distinct callsign (which may be a collective or beacon identifier) heard on the channel, together with node-related routing information, the latest computed route and other miscellaneous information. The table is indexed by node ID (NID), which is used in the computed route and in other tables instead of the awkward callsign string. The link table contains an entry for each distinct (unordered) node pair observed in a monitor header. Each entry includes the from-NID and to-NID of the first instance found, together with link-related routing information and other miscellaneous information. Both tables are dynamically managed using a cache algorithm based on a weighted least-recently-used replacement mechanism described later.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.