Computer Network Routing Based on Imprecise Routing Tables

MILAN TUBA
Megatrend University Belgrade
Faculty of Computer Science
Bulevar umetnosti 29, 11070 Novi Beograd
SERBIA
tuba@ieee.org

Abstract: This paper examines problems that arise from the fact that dynamic routing never relies on the precise, current information about the network state. It is a normal expectation that dynamic routing has to give better results than a static one. However, it takes some time to collect information about the network current state, and optimization is always done with that imprecise information. This situation is examined by a complete mathematical analysis of a simple network. We show that dynamic routing gives better results than static, as expected, but that the margin is much smaller then intuitively expected. Further analysis shows that that minor advantage can easily be lost if there is even a small error in the dynamic routing tables, and actually dynamic routing can easily become worse than static. Quantitative analysis shows that delays in building routing tables can affect dynamic routing performance unexpectedly strongly. The conclusion is that dynamic routing should not try to adapt to traffic changes very fast.

Key–Words: Computer network routing, Static optimization, Dynamic optimization, Modeling

1 Introduction

Optimization is one of the most widely applicable mathematical techniques. Almost any practical problem can be represented as a system where certain function should be minimized (or maximized) under certain conditions. The systems usually change over time and one of the possible classifications of corresponding optimizations is to *static* and *dynamic* cases. Alternative terminology is *fixed* and *adaptive*.

It should be noticed that pure static and pure dynamic optimizations are practically never used. Pure static (fixed) optimization would imply that one solution is used forever. That is very rarely the case, we usually adjust optimization solutions to system changes but relatively rarely, for example once every week or once every hour. On the other side, pure dynamic (adaptive) optimization would adjust to system changes infinitely fast (in time zero) which is impossible. In such optimizations we try automatically to adjust to the system changes as fast as possible. That is the main problem with adaptive optimization. For many problems we can consider that attempt to adjust ”as fast as possible” is also ”good enough”. In this paper we show that in some (many?) cases such assumption can be dangerous and that our intuitive feeling of what is ”good enough” can be very misleading.

This relation between fixed and adaptive optimization is investigated on the computer network routing problem.

2 The Network Design Problem

The network design problem is defined as:

- For given locations of nodes (set of computer sites), traffic matrix (offered traffic for each pair of nodes) and cost matrix (cost to transfer a message for each pair of nodes)
- With performance constraints: delay, throughput, reliability,
- Find values for variables:
  - Topology (which nodes will be connected directly with a link and which will have to communicate indirectly, using other nodes as intermediate stations)
  - Capacity assignment (how much traffic will each link be able to carry)
  - Routing (flow assignment) (which paths messages between any pair of nodes will follow) that
- Minimize the cost (of building and maintaining the whole network) for given delay (time that a message spend in the network)

Other formulations of the problem are: minimize delay for the given cost or maximize throughput for given cost and delay. It has been shown that all these problems are similar and that the same techniques can be applied.

It has to be emphasized in the beginning that this problem is intractable if full and exact solution is required. Networks can have many hundreds of nodes (computers) and many thousands of terminals. Fortunately, experience has shown that network design can be done hierarchically and still be near optimal. An example is a network for a country. First, we can decide where to put trunks between major cities, then connect small cities to nearest major cities, then make local networks inside the cities. This approach allows us to work with networks of at most 50 nodes at a time. This is a great help, but the problem is still intractable. If we have only 10 nodes, there are 45 potential lines that connect different pairs of nodes. Each link can be present or absent. This gives 2^{45} or $3*10^{13}$ different topologies. If we can examine 1000 topologies per second (too fast even for super-computers), it would still require 1000 years to examine all of them. Full-duplex is assumed here, i.e. there is only one link between two nodes. Without that assumption there would be 90 possible links and the problem would be even worse.

2.1 Optimization Methods

Mathematical programming, network flow and queueing theory are used to solve the distributed, packet-switching network design problem. Many algorithms from graph theory are also used. They include Dijkstra's algorithm for the single source shortest paths problem, Floyd's algorithm for the all-pairs shortest paths problem, Prim's algorithm and Kruskal's algorithm for minimum cost spanning tree.

Depending on assumptions on some of the three subproblems, different methods were used:

- Solving for capacity assignment when topology and routing policy are given. If the costs are linear, LaGrange multipliers are good enough. For nonlinear costs, dynamic programming is used.

- Solving for routing when topology and channel capacities are given. The Minimum Link and The Flow Deviation Algorithm are used; both are heuristic.

- Solving for capacity and flow assignment when topology is given is a more general problem. This problem has many local minima and only suboptimal solutions exist. There are algorithms for linear, concave and discrete costs. For topological design, two algorithms, both heuristic, are used. One is The Branch X-Change Method; the other is Concave Branch Elimination.

Another approach is to use graph theory to find the suboptimal topology.A cut between two nodes is a set of arcs whose removal disconnects two nodes. A minimal cut is one in which replacement of any of its members reconnects the graph. There is a theorem that states that the maximum flow between any two arbitrary nodes in any graph cannot exceed the capacity of the minimum cut separating those two nodes. There is the stronger result that maximum flow is equal to the capacity of the minimal cut. This Max-Flow Min-Cut Theorem helps to optimize networks but some systematic way of searching for the minimum cut is needed.

Network design and analysis almost always involve under-determined systems, especially when routing policy has to be determined. The number of possible routings grows with the factorial of the number of the nodes in the networks and the number of possible topologies is exponential in the number of links. The number of constraints (such as "everything that goes in must go out" for each node that is neither source nor sink) is typically polynomial in the number of nodes in the network.

The network design problem with all three subproblems: topology, capacity assignment and routing is intractable. When some of the three problems are fixed or some assumptions are made, the remaining problems may have optimal solution that can be obtained within a reasonable time. Exactly, what assumptions can be made about some of the three subproblems depends on the current state of the technology.

3 The Routing Problem

The routing problem in packet-switched networks is one of the three related problems during the design phase. Routing has also to be calculated later, for dynamic adjustments to the traffic conditions on the network.

The goal of routing is to direct offered load (user traffic) from sources to destinations, along selected routes, in such a way that some parameters are optimized while respecting some restrictions. Objectives include maximizing network performance (minimizing delay and/or maximizing throughput) while minimizing the cost of the network itself (equipment, facilities and maintenance). Besides topology, additional constraints are imposed by the technology that

is used, network services that are provided and user requests. Routing as a multiobjective, multiconstraint optimization problem has been a reach area for research for many years. The evolution of network technologies constantly introduces new context in which older routing technologies have to be reexamined and adjusted.

3.1 Routing Functions

Many different routing algorithms usually have in common the fact that they provide three core routing functions:

- Assembling and distributing network and user traffic state information. This information includes service requirements and current locations of users, services provided by and resources available within the network, and restrictions on use of these services and resources. State information in generating and selecting routes and it can contain not only measured, but also predicted values.
- Generating and selecting feasible and optimal routes based on user and network state information. Feasible are routes that satisfy all constraints while optimal are best with respect to some performance objective. This function is often computationally intensive and may require heuristic approach.
- Forwarding user traffic along selected routes. It can be done in connection-oriented and connectionless way. We are more interested here in connectionless routing.

3.2 Centralized and Distributed Routing

Each of the three mentioned routing functions may be implemented in different degrees of centralized or distributed form. By changing the degree of centralization dynamism, robustness and manageability are affected. Both, centralized and distributed implementations have advantages and disadvantages.

With a centralized implementation, a single node completely determines routing. Such system is easier for management because everything about routing is in the same place and special hardware can be easily added. However, if that node fails, routing is completely halted. Also, dynamic routing is less responsive since it takes more time to collect all the necessary information about the network in a single central node.

With a distributed (decentralized) implementation of the routing, it is possible to have multiple nodes independently replicate routing function, without exchanging information, or to distribute functionality so that each node provide portions of the functionality. In both cases, it is more difficult to manage the routing system, but advantages are more important. The fault tolerance is greatly increased. Response delay is reduced since needed information is closer to places where it needed. The amount of routing resources at any node is decreased and system can can easily incrementally grow. Most routing systems today are distributed.

4 Routing in the Internet

Internet routing algorithms changed over time. They were shifting in the direction of more dynamic routing, but not without problems. In practice they were refined, and theoretical explanations were searched, some of which are in this monograph.

4.1 Routing Algorithms

Routing algorithm should generally have some desirable properties: correctness, simplicity, robustness, stability, fairness, optimality etc. However, global optimality and fairness to individual users are contradictory goals. The optimal routing will be some trade-off between optimality and fairness, and it is necessary to define exact goal of optimization. Minimizing mean packet delay and maximizing total network throughput are in conflict and, as a compromise, the number of hops is often minimized. Reducing the number of hops tends to reduce both, the delay and amount of bandwidth consumed.

As a consequence of the *Optimality Principle* (which states that if router B is on the optimal path from A to C, then the optimal path from B to C also falls along the same route), the set of optimal routes from all sources to a given destination forms a tree rooted at the destination. Such a tree is called sink tree and it is a benchmark against which other routing algorithms can be measured.

4.2 Shortest Path Routing

Representative of static routing algorithms is the *Shortest Path Routing*, implemented by Dijkstra's algorithm. The algorithm is well known and the only thing that remains to be done is to determine distance metric. It is usually the number of hops, but it can be geographic distance, mean queuing or transmission delay. In the most general case distance can be computed as a function of the distance, bandwidth, aver-

age traffic, communication cost, mean queue length, measured delay etc. with appropriate weights.

Another static routing algorithm is *Flooding*, in which every incoming packet is sent out on every outgoing line, except the one it arrived on. It is not practical in most applications.

More advanced static routing algorithm that, besides topology, takes into account load is *Flow-Based Routing*. It is applicable in networks where data flow for each pair of nodes is relatively stable. From mean delays on all the lines, a mean flow for the whole network is calculated. It can be optimized by going from one feasible routing to another that relieves heavily used links.

4.3 Distance Vector Routing

Distance vector routing is older of the dynamic routing algorithms. It is also known as the distributed Bellman-Ford algorithm and Ford-Fulkerson algorithm. It was the original Arpanet routing algorithm since 1969, and later used in the Internet, under the name RIP.

This was distributed, adaptive algorithm that was an ambitious first attempt and source for lot of research, but with some fundamental flaws. It belongs to a class of shortest path algorithms.

Each communication link is assigned a positive number as its length (which can be different in each direction). This length should represent the level of congestion on that link. Each path is a sequence of links, and its length is the sum of the lengths of its links. The Arpanet algorithm tries to send packets along the shortest path between the origin and destination nodes. In such a way it avoids congested links and reduces delays.

Since the length of a link measures traffic congestion, and Arpanet algorithm is adaptive, this length has to be periodically updated. Neighboring nodes exchanged their estimated shortest distances to each destination every 625 milliseconds. Each link length is dependant on the number of the packets waiting in the link queue.

Link lengths changed rapidly, reflecting statistical traffic changes and effects of routing updates. To stabilize oscillations, a large positive constant was added to the link lengths. This effort reduced sensitivity of the algorithm to traffic congestion without completely removing oscillations. It has been noticed that there is a tendency to route everything through less utilized parts of the network, which in turn caused that part to become heavily utilized. Everything then was routed through other part of the network and that part would become heavily utilized and so on, back and forth. For several years this caused problems and finally, that algorithm had to be replaced.

Each router maintains a routing table with one entry for each router in the network. This entry contains two parts: preferred outgoing link for that destination and estimated distance to that destination. As before, distance can be measured as number of hops, delay in milliseconds etc. Each router independently learns the distance to each of its neighbors. For number of hops it is trivially 1, delay can be measured by sending special ECHO packets. Periodically, each router sends that routing table to each of its neighbors. With that information each node calculates the complete routing table.

The problem with distance vector routing is slow convergence, especially in case of bad news. When some link fails, that information is propagated very slowly. There were many attempts to overcome that problem, for example the Split Horizon algorithm, but none was good enough.

4.4 Link State Routing

Link state routing was a version of the dynamic routing algorithms which replaced the original Arpanet algorithm in 1979. Besides the problem of slow convergence, it also solves the problem of not considering the bandwidth, both present in the older distance vector algorithm.

The complete topology and all delays are experimentally measured and distributed to each other router. In such a way all relevant factors must have been taken into account.

The router learns about its neighbors and estimates the delay by sending ECHO packets which require immediate response. If queuing delays are included in the delay estimate, the estimate will be better, but there is a danger that the routing will oscillate between alternate paths. Link state packets can be built periodically, or only when something significant occurs.

The length of each link is calculated by using delays for each packet that crosses that link. Each link length is updated every 10 seconds and new value is average (including queuing and propagation times) during the preceding 10 seconds. These link lengths are broadcasted at least once every 60 seconds by using a flooding algorithm, modified with sequence numbers and packet ages. The algorithm is asynchronous and this improved its stability.

When all the information is accumulated, for each link two values are computed, one for each direction, and used separately or averaged. Dijkstra's algorithm is then run locally to find shortest paths to all destinations.

4.5 OSPF - The Interior Gateway Routing Protocol

The latest version of the Internet routing protocol is the OSPF (Open Shortest Path First), also a variant of the link state algorithm. It was introduced in 1990, and defined in RFC 1247.

Having experience with problems with previous algorithms, the following requirements had to be met:

- It had to be "open" i.e. published in the open literature, not proprietary.
- Support for different distance metrics: physical, delay etc.
- Dynamic algorithm that adapts to changes in the topology automatically and quickly.
- Routing based on type of service (care about real-time traffic).
- Bifurcated routing, splitting single stream of packet along multiple routes, in appropriate percentage.
- Support for hierarchical system. By 1990 it was impossible for any router to know the entire topology.
- Security

OSPF supports three kinds of networks: point-to point between exactly two routers, multiaccess with broadcastings (most LANs) and mutiaccess without broadcasting (most packet-switched WANs). It constructs a graph of the whole network where routers, networks and lines play the main role, not the hosts. For each type of service it maintains separate graph with separate distance function. ASes are internally divided into areas and routers can be internal (in one area), area border, backbone and AS boundary. Information is exchanged between adjacent routers, which is not the same as neighboring (on LANs only one router is elected as designated router).

When a router boots it sends HELLO messages on all of its point-to-point lines and broadcasts them on LANs. After that each router periodically floods LINK STATE UPDATE massages and constructs the graph for its area and computes the shortest path.

4.6 BGP - The Exterior Gateway Routing Protocol

While OSPF is recommended within a single AS, between ASes BGP (Border Gateway Protocol) is used. It is defined in RFC 1654 and partially described in RFC 1268. The main difference between the OSPF and BGP is that an interior gateway protocol has only to move packets as efficiently as possible, while exterior gateway protocol routers have to worry about politics. Different ASes are independent units and they have to decide whose packets will they accept, whose packet will they forward, who has to pay for that etc. Such policies are manually configured into each BGP router.

From the point of view of a BGP router, the world consists of other BGP routers and the lines connecting them. Networks can be stub networks with only one connection to the BGP graph, multiconnected networks that could be used for transit traffic and transit networks (such as backbones) that are primarily used for transit traffic.

BGP is distance vector protocol but with significant modifications. Each BGP router maintains not just the cost to each destination, but the exact path. This prevents the count-to-infinity problem.

5 Problems with Dynamic Routing

Highly dynamic optimal routing has been used in the Internet [1], [2], [3]. Expectations that it will give much better results were not completely fulfilled, because unexpected delays occurred often. Here and in [4], [5], [6], [7], [8], [9] is presented an attempt to give some theoretical explanation for such behavior. Today, this problem is again interesting but in the context of wireless ad hoc mobile networks [10], [11], sometimes using evolutionary computing [12].

Here, for the routing problem we also have accepted terminology to classify routing as static versus dynamic or fixed versus adaptive. In practice, however there is not a clear-cut between the two: fixed or static routing is not fixed forever and dynamic or adaptive is not infinitely fast in its adjustment to the situation on the network. As mentioned before, fixed routing has to accommodate for occasional link failures at least, and adaptive routing needs some time to collect and analyze the current traffic on the network. In reality, routing adjustments are made, the question is how often. In can be done on a daily basis, or every hour, every minute or every few seconds. If it is done every few minutes, it can be called fixed if compared to adjustments every few seconds, or it can be called adaptive if compared to adjustments every few days. Often terms like "semi-adaptive", "semi-fixed", "highly-dynamic" etc. are used.

There are problems with distributed optimal routing algorithms. Kleinrock was the first [13] to point out that "... uncontrolled alternate routing in a congested network can lead to chaos. Indeed, the tele-

phone company tends to limit (and even prohibit completely) alternate routing on unusually busy days (Mother's Day, for example)."

It takes significant time to calculate new routing tables, both to accumulate data in any node, and to exchange data among nodes. By the time the calculation is finished load may be sufficiently different to make the tables obsolete, and the routing far from optimal.

By working on the knees of sharply rising delay curves, highly dynamic optimal routing can expend massive amounts of network resources for no benefit. It will be shown that congestion can be avoided in a useful range of cases by quasi-static bifurcated routing with conservative load estimates, and that the delay penalties for use of this, less then optimal, routing are small. There are cases where dynamic routing can offer significant performance improvements, but without full load information and without infinitesimal route calculation time the game theoretic "maximal loss" is not minimized.

A complete mathematical analysis of a simple network will be done. It will show that dynamic routing offers an improvement over static routing that is smaller than expected. That minor theoretical gain can easily be lost, and situation can actually become worse, if there is even a small error in the dynamic routing tables. An interesting and somewhat surprising solution is offered. Congestions can be avoided if optimization is not tried too hard. Dynamic routing is good, but only if the tables can be recalculated very quickly. Static routing is better then attempted, but unsuccessful optimal dynamic routing.

6 Mathematical Model

A simple three node network and even simpler offered load will be examined. Nodes are A, B and C, and all traffic is from A to B. There are two possible different routes: a direct path from A to B, and an indirect path, of the length two, that goes through C. Let us assume that all three lines are of the same capacity μ bits per second. Our routing problem is then reduced to making a decision about what fraction α of the total offered traffic will be sent along the indirect path of the length two. The remaining fraction $1-\alpha$ of the total load will be send along the direct path. Let us call α a branching coefficient. Let offered load be λ bits per second and ρ will, as usually, denote utilization λ/μ. We will also assume a Poison input stream of messages, and an exponential service time on lines.

There are some limitations for the parameters that we introduced [4], [9]. Parameter α is a fraction (probability) so we certainly have $0 \leq \alpha \leq 1$. For this particular case, there is an even stronger condition. We may have to send some traffic along the longer route, which is more expensive, has longer wait time etc., only if the direct path is overloaded (whatever the definition of the "overload" is). It is obvious, however, that it never pays off to send more traffic along the indirect route than along the direct route. If the lines were of different capacities, costs, reliabilities etc., this would not have to be the case, but according to our assumptions, we get the limitation that reasonable interval for α is $0 \leq \alpha \leq 0.5$ (this will formally follow from the requirement that utilization for each line must be less than 1).

There may be some additional limitations for α. If the total offered load λ is less than the line capacity μ, then there are no problems. The network, however, may withstand the total offered load of $\lambda < 2\mu$ or $\rho < 2$. The reason for this is that we have two alternative paths, each of the capacity μ. It is obvious that when the total load approaches 2μ, there is no more freedom in selecting α. It has to be equal to 0.5, or one path will become overloaded, introducing infinite delays.

The new set of limitations for α can be calculated as follows. With the total load λ, line capacity μ, utilization ρ, and the branching coefficient α, the utilizations of the direct path ρ_1, and the utilization of the indirect path ρ_2 will be:

$$\rho_1 = (1-\alpha)\rho \quad \text{and} \quad \rho_2 = \alpha\rho \tag{1}$$

In order to keep the network in a stable state (to avoid infinite queues and delays), we have to avoid overloading any of the two paths. By solving $\rho_1 < 1$ and $\rho_2 < 1$, we get an additional constraints $\alpha > 1 - 1/\rho$ and $\alpha < 1/\rho$. If we check the second constraint, we see that it is completely included in the previous constraint $\alpha < 0.5$.

Then, the final set of constraints is:

$$\mu > 0 \tag{2}$$

$$0 \leq \rho < 2 \tag{3}$$

$$\max\left(0,\ 1-\frac{1}{\rho}\right) \leq \alpha \leq 0.5 \tag{4}$$

The left constraint in the last expression is different from zero for $\rho > 1$.

6.1 Optimal Waiting Time

The waiting time (including service time) for an M/M/1 queuing system is:

$$W_{M/M/1}(\rho) = \frac{1}{\mu(1-\rho)} \tag{5}$$

$W_{M/M/1}$ is a function of λ and μ, but they are connected through ρ, and μ can be considered constant.

By using Kleinrock's Independence Assumption, the total waiting time for our network is:

$$W(\alpha,\rho) = (1-\alpha)W_{M/M/1}(\rho_1) + 2\alpha W_{M/M/1}(\rho_2) \quad (6)$$

By substituting Equations (1) and (5), we get

$$W(\alpha,\rho) = \frac{1-\alpha}{\mu[1-(1-\alpha)\rho]} + \frac{2\alpha}{\mu[1-\alpha\rho]} \quad (7)$$

or

$$W(\alpha,\rho) = \frac{3\rho\alpha^2+(1-3\rho)\alpha+1}{\mu[1-\alpha\rho][1-(1-\alpha)\rho]} \quad (8)$$

Our goal is to optimize the waiting time so we need a derivative. Parameter under our control is α. Differentiation gives:

$$\frac{dW(\alpha,\rho)}{d\alpha} = \frac{\rho^2\alpha^2 - 2\rho(2\rho-3)\alpha + 2\rho(\rho-2)+1}{\mu(1-\alpha\rho)^2[1-(1-\alpha)\rho]^2} \quad (9)$$

The optimal (minimal) waiting time, when branching probability α is selected optimally

$$W_{opt} = \frac{(21-18\sqrt{2})\rho+32\sqrt{2}-45}{\mu\rho[(6\sqrt{2}-7)\rho-12\sqrt{2}+14]} \quad (10)$$

7 Changing Offered Load

Previous chapter assumes that we know that the offered load is λ exactly, and that it does not change in time. This case is not really interesting. In reality, offered load is always changing in time, and that is what makes difference between static and dynamic routing, but also gives possibility for an error when calculating dynamic routing tables.

7.1 Uniformly Changing Offered Load

Let us consider more general and more realistic case, when the offered load changes in time between the lower limit l and the upper limit h, where $0.3 \leq l < h < 2$ must be satisfied. To make calculations easier (or possible) we will assume that the load changes uniformly. That means that the value for the offered load spends equal amount of time inside any subinterval of the same size, included between l and h. Such distribution corresponds, for example, to constant-speed load shift from l to h, back and forth. This assumption that load changes uniformly between l and h is somewhat artificial, but not very far from what really happens in the network.

7.2 Optimal Dynamic Routing

We will now calculate the waiting time for optimal dynamic routing. We select our optimal branching probability α infinitely fast, and at any moment it follows precisely the changing load λ. The total waiting time will be expected value with regard to the distribution $g(\rho)$ of the changing load:

$$W_{opt\_dyn} = \int_l^h W_{opt}(\rho)\, g(\rho)\, d\rho \quad (11)$$

By substituting Equation (10) and $g(\rho)$ for uniform distribution, we get

$$W_{opt\_dyn} = \frac{1}{h-l}\int_l^h \frac{(9\sqrt{2}-12)\rho-17\sqrt{2}+24}{\mu(3\sqrt{2}-4)\rho(2-\rho)}d\rho \quad (12)$$

By solving this integral, we get the best we can hope for in the case of uniformly changing load. Optimal dynamic routing gives waiting time:

$$W_{opt\_dyn} = \frac{(24-17\sqrt{2})\,\ln\left(\frac{h}{l}\right)+\sqrt{2}\,\ln\left(\frac{2-l}{2-h}\right)}{2\mu\,(3\sqrt{2}-4)(h-l)} \quad (13)$$

7.3 Optimal Static Routing

Let us now examine static routing where the branching probability α will always have the same, fixed value. To find the optimal value for that fixed branching probability α, we do again differentiation and integration, but in the reverse order. Previously, we differentiated W to find optimal α for a particular ρ and then, using that optimal α, integrated over all possible values for ρ (with regard to distribution for ρ). Now, we will integrate over all possible values for ρ (assuming that α is fixed) to find average W and then differentiate that expression with respect to α to find the optimal fixed value for α, which minimizes W.

Average waiting time for a fixed α will be:

$$W_{avg} = \int_l^h W(\rho)\, g(\rho)\, d\rho \quad (14)$$

or, after we substitute Equation (7) and $g(\rho)$ for uniform distribution

$$W_{avg} = \frac{1}{h-l}\int_l^h \left[\frac{1-\alpha}{\mu[1-(1-\alpha)\rho]} + \frac{2\alpha}{\mu(1-\alpha\rho)}\right] d\rho \quad (15)$$

By solving this, we get

$$W_{avg} = \frac{1}{\mu(h-l)} \ln \frac{(1-\alpha l)^2[1-(1-\alpha)l]}{(1-\alpha h)^2[1-(1-\alpha)h]} \quad (16)$$

Now, we differentiate this expression with regard to α:

$$\frac{dW_{avg}}{d\alpha} = \quad (17)$$

$$\frac{3\alpha l - 2l + \alpha^2 hl - 4\alpha hl + 2hl + 1 + 3\alpha h - 2h}{\mu\,(1-h+\alpha h)\,(-1+\alpha h)\,(1-l+\alpha l)\,(-1+\alpha l)}$$

we get an expression for the optimal waiting time for static routing:

$$W_{opt\_stat} = \frac{\ln\left(\frac{l^3[R+h(4l-5)-3l]^2[R+h(2l-1)-3l]}{h^3[R+h(4l-3)-5l]^2[R+h(2l-3)-l]}\right)}{\mu(h-l)} \quad (18)$$

This case represents pure static routing if the boundaries l and h are fixed and never change. In practice, we use a quasi-static routing where the boundaries l and h do change over time, but much slower than the offered load ρ. We adjust l and h, and corresponding α_{opt_stat}, but we do it once every hour or so. For shorter periods of time routing is static, while dynamic routing chases changing offered load continuously.

7.4 Comparison

Now, we will compare optimal dynamic routing and optimal static routing. Formula that is used to calculate improvement is

$$\text{Improvement} = \left(\frac{W_{opt\_stat}}{W_{opt\_dyn}} - 1\right) * 100\% \quad (19)$$

The following Table 1 shows improvement in percents (reduction of delays) when optimal static routing is replaced by optimal dynamic routing, for different intervals $[l, h]$, where offered load ρ is uniformly changing.

l,h	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.3	0.6	1.3	1.8	2.2	2.5	2.7	2.8	2.9
0.5		0.2	0.6	1.0	1.3	1.7	2.0	2.3
0.7			0.1	0.4	0.7	1.1	1.5	2.0
0.9				0.1	0.3	0.6	1.0	1.6
1.1					0.1	0.3	0.7	1.3
1.3						0.1	0.4	1.1
1.5							0.1	0.7
1.7								0.4

Table 1: Dynamic vs. Static routing, improvement in percents

Rows in the table give corresponding improvement for particular l, columns for h. Since $l < h$, only the upper right triangle of the table is used, diagonal excluded. First impression is surprisingly small improvement that dynamic routing introduces. It allows us to make claim that too zealous optimization is harmful. Even without any errors in calculating routing tables, best improvement we can hope for, the upper limit, is given in Table 1. Average improvement is about 1%, maximal improvement is less than 3%. It is not surprising that maximal improvement is achieved when interval $[l, h]$ is wide. Traffic then varies a lot, and if we can follow that wide variations, improvement will be more significant.

When we look at this modest improvement, we should keep in mind that we are dealing with a very simple model with only three nodes and one source. In a larger network, it is possible that improvement would be better, but chances for an error in the dynamic routing tables would also be better. The combined effect would probably be the same.

The conclusion is that optimal dynamic routing gives modest improvement over optimal static routing. That small improvement can easily be annihilated, and actually dynamic routing can give larger delays than static, if there are any errors in the dynamic routing tables. Such errors always exist, because it takes significant time to calculate new routing tables, both to accumulate data in any node, and to exchange data among nodes. By the time the calculation is finished, load may be sufficiently different to make the tables obsolete, and the routing far from optimal.

8 Imprecise Adaptive Routing

Now, we show how that small advantage can be lost and why dynamic routing can become worse than static, even for relatively minor errors in traffic estimate.

The goal in this section is to quantitatively examine how imprecise (obsolete) traffic information used

for calculating dynamic routing affects delays in the network and when dynamic routing becomes impractical because optimal static routing becomes better.

We need a simple, but not far from the reality, mathematical model to represent obsolete traffic information. We have already made assumption that offered load for the network changes uniformly between l i h. We can add another assumption that uniform change is by constant speed from l to h and back and so forth. Time delay in collecting traffic information can then be represented by fixed underestimate of the traffic (or, for the other direction, by fixed overestimate). That practically means that underestimated value $\rho - d$ should be substituted for ρ in the expression for optimal value for branching coefficient α.

$$\alpha_{opt\_d} = \frac{2\rho - 2d - 3 - \sqrt{2}\rho + \sqrt{2}d + 2\sqrt{2}}{\rho - d} \quad (20)$$

8.1 Optimal Imprecise Dynamic Routing

This imprecise α_{opt_d} we substitute into expression for waiting time

$$W_{opt\_d} = \frac{1 + 3\frac{A\rho}{d-\rho} - \frac{A}{d-\rho} + 3\frac{A^2\rho}{(d-\rho)^2}}{\mu\left(1 - \rho - \frac{A\rho}{d-\rho}\right)\left(1 + \frac{A\rho}{d-\rho}\right)} \quad (21)$$

where

$$A = (2 - \sqrt{2})(\rho - d) + 2\sqrt{2} - 3 \quad (22)$$

Now, we can calculate, as before, optimal waiting time for the network when offered load uniformly changes from l to h:

$$W_{opt\_dyn\_d} = \frac{1}{h - l}\int_l^h W_{optd}(\rho)d\rho \quad (23)$$

Calculating this integral gives a complicated expression that can be written in parts:

$$W_{opt\_dyn\_d} = \frac{1}{\mu\sqrt{a_1 * a_2}(h-l)2(3-2\sqrt{2})^2} * \quad (24)$$

$$* \Bigg\{ \sqrt{a_1 * a_2}\left[(17 - 12\sqrt{2})\ln\left(\frac{b(l) + \sqrt{2}d}{b(h) + \sqrt{2}d}\right) + \right.$$

$$\left. (34 - 24\sqrt{2})\ln\left(\frac{2b(l) + \sqrt{2}d}{2b(h) + \sqrt{2}d}\right)\right] +$$

$$+4\sqrt{a_1}(a_3 - a_6)(29d\sqrt{2} - 41d + 41\sqrt{2} - 58) +$$

$$+2\sqrt{a_2}(a_5 - a_4)(29d\sqrt{2} - 41d + 82\sqrt{2} - 116)\Big\}$$

where

$$a_1 = (2\sqrt{2} - 3)(d^2 + 4) + 2(6\sqrt{2} - 8) \quad (25)$$

$$a_2 = (2\sqrt{2} - 3)(d^2 + 4) + (6\sqrt{2} - 8) \quad (26)$$

$$a_3 = \arctan\left(\frac{(\sqrt{2} - 1)(d - 2l + 2)}{\sqrt{a_2}}\right) \quad (27)$$

$$a_4 = \arctan\left(\frac{(1 - \sqrt{2})(d - 2l + 2)}{\sqrt{a_1}}\right) \quad (28)$$

$$a_5 = \arctan\left(\frac{(1 - \sqrt{2})(d - 2h + 2)}{\sqrt{a_1}}\right) \quad (29)$$

$$a_6 = \arctan\left(\frac{(\sqrt{2} - 1)(d - 2h + 2)}{\sqrt{a_2}}\right) \quad (30)$$

$$b(t) = t^2 - (2 + d)t + d \quad (31)$$

This formula for $d = 0$ reduces to formula for waiting time for optimal dynamic routing, Equation (13), that we had before.

8.2 Static and Imprecise Dynamic Routing

Now, we can compare imprecise dynamic routing with optimal static routing. As before, we will calculate improvement in percents (reduced delays) when optimal static routing is replaced by imprecise dynamic routing. Improvement is calculated for different intervals $[l, h]$ where offered load uniformly changes. When d becomes large enough, improvement will become negative, i.e. static routing will become superior to this sufficiently imprecise dynamic routing.

l,h	0.7	0.9	1.1	1.3	1.5	1.7	1.9
0.5	-2.4	-1.3	-0.5	0.1	0.6	0.9	-0.2
0.7		-1.1	-0.6	-0.2	0.2	0.5	-0.6
0.9			-0.7	-0.4	-0.1	0.1	-1.0
1.1				-0.6	-0.4	-0.2	-1.5
1.3					-0.6	-0.6	-2.0
1.5						-1.0	-2.8
1.7							-4.2

Table 2. Improvements in percents for $d = 0.15$

For $d = 0.15$ almost all elements are negative, which means that is better not to try dynamic routing with that size of error in traffic information. Static routing is better and error is only 7.5% of the total capacity.

9 Conclusion

Previous tables give an interesting and somewhat surprising solution. Congestions can be avoided if we do not try to optimize too hard. Dynamic routing is good, but only if we can recalculate tables very fast. Static routing is better than attempted, but unsuccessful optimal dynamic routing.

Acknowledgment: The research was supported by the Ministry of Science, Republic of Serbia, Project No. 144007

References:

[1] Mills, D. L.: Exterior Gateway Protocol formal specification, *RFC 904*, 1984

[2] Hedrick, C.: Routing Information Protocol, *RFC 1058*, 1988

[3] Blokzijl, R.: RIPE Terms of Reference, *RFC 1181*, 1990

[4] Tuba, Milan: Optimal Bifurcated Routing in Computer Networks, *Proceedings of the 38th Conference ETRAN'94*, Vol. 3, pp. 55-56, Niš, 1994

[5] Tuba, Milan: Dynamic Routing in Computer Networks Introduces Small Improvements, *"Technology of Programming" Conference*, pp. 7, Matematički institut, Beograd, 1994

[6] Tuba, Milan: Conditions when Dynamic Routing Becomes Worse than Static, *PriM 94, IX Conference on Applied Mathematics*, pp. 81-82, Budva, 1994

[7] Tuba, Milan: Cost Function for Communication Links in Computer Networks, *Bulletins for Applied Mathematics (BAM)*, LXXIII-1028/94, pp. 115-122, Budapest, 1994

[8] Tuba, Milan: Dynamic Routing Based on Obsolete Traffic Information, *Proceedings of the International Conference on Technical Informatics ConTI'94*, Vol. 4, pp.103-110, Timisoara, 1994

[9] Tuba, Milan: A Mathematical Model for Routing Comparison in Computer Networks, *Bulletins for Applied Mathematics (BAM)*, LXXXVI-A 1565/98, Arad, July 1998, pp. 493-503

[10] Karavetsios, P., Economides, A.: Performance Comparison of Distributed Routing Algorithms in Ad Hoc Mobile Networks, *WSEAS Transactions on Communications*, Vol. 3, Issue 1, 2004, pp. 317-321

[11] Sokullu, R., Karaca, O.: Comparative Performance Study of ADMR and ODMPR in the Context of Mobile Ad Hoc Networks and Wireless Sensor Networks, *International Journal of Communications*, Issue 1, Volume 2, 2008, pp. 45-53

[12] Kumar, D., Bhuvaneswaran, R.: ALRP: Scalability Study of Ant Based Local Repair Routing Protocol for Mobile Ad Hoc Networks, *WSEAS Transactions on Computer Research*, Vol. 3, Issue 4, Apr 2008, pp. 224-233

[13] Kleinrock, L. *Computer Networks*, in "Computer Science", ed. A. F. Cardenas, L. Presser & M. A. Martin, pp. 241-284, Wiley-Interscience, New York, 1972