

A Delay Constrained Minimum Hop Distributed Routing Algorithm using Adaptive Path Prediction

A. R. Mohd Shariff

University of Bradford/Department of Computing, Bradford, United Kingdom
a.r.mohdshariff@bradford.ac.uk

M. E. Woodward

University of Bradford/Department of Computing, Bradford, United Kingdom
m.e.woodward@bradford.ac.uk

Abstract— We propose a new delay-constrained minimum-hop distributed routing algorithm (DCMH) for unicast-routing. DCMH uses a novel path feasibility prediction (PFP) model that is able to predict the feasibility of a tentative routing path, and employs a crankback facility. It is a hybrid algorithm where it uniquely integrates the use of local-state and the global binary connectivity information maintained at each node to compute routing paths. Simulation results show that average message overhead is modest with good connection success and low nodal storage.

Index Terms—distributed algorithm, routing, quality of service, path feasibility prediction

I. INTRODUCTION

The pervasive demands for connection-oriented real-time multimedia applications are ever blooming. These applications are bandwidth hungry, impose strict delay guarantees, stable jitter and low packet loss probabilities, which constitutes the notion of Quality-of-Service (QoS). One major element in the provisioning of QoS services in the Internet is QoS-routing [1], [19]. QoS-routing algorithms can dynamically determine a feasible path based on certain constraint optimizations of a requested flow. Two most important schemes studied are source routing and distributed routing. They exhibit performance trade-offs in terms of computational and protocol overhead. In source routing where path computation is done at the source node, an updated ‘global link-state database’ of the network is kept at each node using link-state protocols (or distance vector). Protocol overhead is expensive due to frequent triggering of link-state update packets especially when a network is large and traffic is dynamic. For low call-blocking probability, the global-state maintained must be accurate, particularly for delay-constrained routing [3]. There is a clear trade-off between

protocol overhead and impreciseness of routing information, influenced by different triggering policies for network state updates [20], [22]. In addition, the storage size of the global-state database increases exponentially with network size and the number of entries, each representing a different QoS metric.

In distributed routing, path computation is distributed amongst intermediate routers between source and destination. Distributed routing is categorized into two techniques. First, routing that relies on the global-state and second, routing that depends only on the local-state. Distributed routing algorithms (DRAs) based on the global-state require each node to compute a forwarding table using Dijkstra’s algorithm (or Bellman-Ford) with entry specifying the next-best hop to the destination. A routing decision is made locally at each node on a hop-by-hop basis. Although computationally efficient, these algorithms inherit the same problem of state information imprecision and high protocol overhead as in source routing. Flooding [1] is the most studied distributed routing scheme that does not require the global-state. A connection-request packet is flooded entirely or selectively [18] in the network where it traverses on a link-by-link basis whilst QoS requirement is met until it reaches the destination, or is dropped at the particular link where it is violated. Same connection-request packets taking different paths may arrive at an intermediate node or the destination especially in lightly loaded network conditions. Pruning of redundant paths are necessary for the ‘breadth-first-search’ schemes of flooding. One limitation of DRAs is the computational burden (i.e. message overheads) of finding optimal paths for multiple additive constraints (e.g. a delay-constrained least-cost problem), particularly if the DRA is dependent on the local-state alone. On the contrary, computing optimal paths using source routing algorithms (SRA) for the same problem is found to be NP-complete. Although a study suggested that NP-completeness is rarely found in practical except for four conditions [21], the persistent problem is maintaining an updated global-state for exact (loop-less) QoS-routing. We strongly believe that DRAs

Based on “A Distributed Algorithm for Unicast QoS-Routing using Path Feasibility Prediction”, by A. R. Mohd Shariff, and M. E. Woodward which appeared in the Proceedings of the IEEE International Workshop on High Performance Switching and Routing 2006, Poznan, Poland, June 2006. © 2006 IEEE.

are holistically scalable compared to SRAs when performance measures of protocol overhead, nodal storage and scalability are considered. A DRA that is able to compute a delay-constrained minimum cost path with modest overhead and using minimum state information can be an attractive solution for QoS-routing in the Internet.

We present in this paper a new DRA for unicast routing called Delay Constrained Minimum Hop (DCMH). We consider DCMH as a special case of Delay Constrained Least Cost (DCLC) problem where DCMH computes a delay-constrained path with minimum hop in a distributed fashion. We developed a novel inequality model called the Path Feasibility Prediction (PFP) that is adaptive to path delay variations where it allows DCMH to predict and make an ‘a priori’ decision as to whether a link should be accepted or otherwise. The inequality model of the PFP combines pre-computed Shortest Path (SP) hop count information to all nodes as potential destinations and on-demand local-state into a single function. DCMH is a ‘guided depth-first-search’ algorithm as the DRA is always guided towards reaching the destination by the SP hop counts. DCMH uses a crankback procedure to retrace to a good path when a path delay violation is detected. The concept for DCMH was originally devised from a sequential decoding technique introduced by R.M.Fano [12]. This paper is an extension of [2] where the basic PFP model and some general results were presented. In this paper, we provide a detailed analysis of the properties and characteristics of the proposed PFP model. We comment on its strengths and limitations, and practicality for QoS-routing. We present two new results from the simulation for the performance measures of inefficiency and computed path delay variation.

The rest of this paper is organized as follows. An overview of the sequential decoding procedure of Fano is given in Section II. In Section III, a thorough explanation of the PFP model is presented focusing on its properties and characteristics, and path adaptability. The routing table structure and information of DCMH is described in section IV. The routing control message requirement is presented in Section V. We explain thoroughly DCMH heuristics and discuss its complexities in section VI. Simulation results are given and discussed in section VII. Finally, section VIII concludes the paper.

II. THE SEQUENTIAL DECODING OF FANO

One of many issues of QoS-routing is scalability. For a DRA, scalability is tantamount to computing QoS-based paths with modest state information, computational (i.e. message) and storage overheads. With no global-state dependency, the combination of an intelligent routing computation model and efficient heuristics is crucial. There exists a sequential decoding procedure for error control coding that is somewhat similar in the context of path computation and performance objectives to QoS-routing. Sequential decoding has a primary advantage over maximum likelihood decoding of Viterbi [13] where it can decode convolutional codes with large constraint

length with arbitrarily small error probability. Of particular interest is the storage free, sub-optimum but computationally efficient sequential decoding algorithm of Fano [5], [12]. For QoS-routing, this corresponds to computing QoS constrained paths using only the local-state with minimal overheads. Fano’s algorithm examines a sequence of nodes in the 2^k code-tree from the start node to the end node making decisions to move forward or backward depending on the Fano ‘Tilted Distance’ (TD) denoted by (1). The TD is the difference in measure between the Hamming distance $d[c(\ell), r(\ell)]$ and $pn\ell$ detected by the decoder at any ℓ^{th} level in the 2^k code tree. The behaviour of $\lambda(\ell)$ fundamentally indicates whether the Fano decoder is tracing a correct or an incorrect path, which is the $c(\ell)$ path. The Hamming distance $d[c(\ell), r(\ell)]$ is the total difference in bits between the transmitted code sequence $c(\ell)$ and received code sequence $r(\ell)$ of the ℓ^{th} level in the 2^k code-tree, p is a design parameter chosen between the Binary Symmetric Channel (BSC) probability p_0 and $1/2$ i.e. $p_0 < p < 1/2$, n is the output bits in a block of the convolutional code and ℓ is the branch level in the 2^k code-tree.

$$\lambda(\ell) = pn\ell - d[c(\ell), r(\ell)] \quad (1)$$

A correct path (i.e. $c(\ell)$) is detected by the decoder when $\lambda(\ell)$ increases as ℓ increases. This implies that the Hamming distance $d[c(\ell), r(\ell)]$ is zero and non-decreasing. If $c(\ell)$ is an incorrect path, then $\lambda(\ell)$ will decrease with increasing ℓ because of the growth in the Hamming distance $d[c(\ell), r(\ell)]$. When an incorrect path is detected, the decoder must locate the node where the correct path diverges. Fano’s algorithm provides a simple yet intelligent search procedure for the node of divergence. This is another key strategy of Fano’s algorithm in addition to the TD. A λ -metric is assigned to each ℓ^{th} node, which is equal to the TD (i.e. $\lambda(\ell)$) of the entire path from the start node to that ℓ^{th} node in the 2^k code tree. The decoder compares this metric against a sequence of defined thresholds $T = k\Delta t$ along the tentative computed path. It either moves a branch forward to a $(\ell + 1)^{th}$ node or backwards to a $(\ell - 1)^{th}$ node depending on whether the value of the metric at that current ℓ^{th} node is larger or smaller than the threshold T . These are the two main concepts (i.e. the TD and forward/backward movement) adopted in the proposed DCMH [2].

III. THE PATH FEASIBILITY PREDICTION (PFP) MODEL

A point-to-point network G is represented as a collection of nodes V connected by directed links E denoted as $G(V, E)$. Each directed link $e_{mn} \in E$ from node v_m to node v_n where $v_m, v_n \in V$ has an associated non-negative delay $d(e_{mn})$ and cost $c(e_{mn}) : E \rightarrow R^+$ and cost $c(e_{mn}) : E \rightarrow R^+$. A routing path from a source v_s to a destination v_d , $v_s, v_d \in V$ is defined as an alternating sequence of nodes and links $P(v_s, v_d) = (v_{s=0}, v_1, v_2, \dots, v_d, e_{s1=01}, e_{12}, e_{23}, \dots, e_{id})$ that can also be expressed either as; a subset of directed links $E(v_s, v_d) = (e_{s1=01}, e_{12}, e_{23}, \dots, e_{id})$ or a subset of nodes $V(v_s, v_d) = (v_{s=0}, v_1, v_2, \dots, v_d)$.

The end-to-end delay of the routing path is the sum of all link delays in $P(v_s, v_d) = P_{sd}$ and is given as

$$D(P_{sd}) = \sum_{e_{mn} \in P_{sd}} d(e_{mn}) \quad (2)$$

Given a delay constraint Δ , the problem of delay-constrained QoS-routing is to search for a suitable path where the sum of link delays does not exceed Δ . Assume that such a routing path exists is the P_{sd} path.

$$D(P_{sd}) = \sum_{e_{mn} \in P_{sd}} d(e_{mn}) \leq \Delta \quad (3)$$

Equation (3) indicates that at any node v_j where the routing is currently located, the accumulated delay so far when subtracted from Δ yields the residual-QoS delay to reach destination v_d from that node v_j . The residual-QoS delay is denoted by $r_{j(d)}$ as indicated in (4).

$$\Delta - \sum_{e_{mn} \in P_{sj}} d(e_{mn}) = r_{j(d)} \quad (4)$$

Each node must know the cost of the least-cost-path (LCP) from itself via each of its adjacent neighbors to v_d . This LCP value to reach v_d from any node v_j via link e_{jk} (or adjacent neighbor v_k) is denoted as $C_{jk(d)}$.

$$C_{jk(d)} = \min_{e_{mn} \in P_{jk(d)}} c(e_{mn}) \quad (5)$$

Given $C_{jk(d)}$ (5) and (4) is known, a basic inequality in (6) is defined where the left-hand term is a predictor and the right-hand term is the residual-QoS delay. Variable w is the predictor multiplier fundamentally included such that (6) holds. The multiplier w takes a single non-negative link delay value at any node in P_{sd} , [$w = d(e_{mn})$] : $E(V_s, V_d) \rightarrow R^+$.

$$C_{jk(d)}^{predictor} \times w \leq \sum_{e_{mn} \in P_{sj}}^{residual-QoS} d(e_{mn}) \quad (6)$$

Rearranging (6) into (7) and expressed as a function of $d(e_{mn})$, three properties are attributed to (8).

$$f(w) = \left(\Delta - \sum_{e_{mn} \in P_{sj}} d(e_{mn}) \right) - (C_{jk(d)} \times w) \geq 0 \quad (7)$$

Note that $f(w) = f(d(e_{mn}))$ since [$w = d(e_{mn})$] : $E(V_s, V_d) \rightarrow R^+$

$$f(d(e_{mn})) = \left(\Delta - \sum_{e_{mn} \in P_{sj}} d(e_{mn}) \right) - (C_{jk(d)} \times w) \geq 0 \quad (8)$$

Property (1)

The function $f(d(e_{mn}))$ provides a general solution for the DCLC problem. The function $f(d(e_{mn}))$ is a delay maximization function and an optimal solution is attainable by optimizing $(C_{jk(d)} \times w)$. Assume $\Omega = (P_1, P_2, \dots, P_n)$ is the set of routing paths from v_s to v_d whose

end-to-end delay satisfy Δ . There exists a path $P_i \in \Omega$ such that the total costs is optimum (i.e. $\min_{P_i \in \Omega} C(P_i)$).

At each node in P_i , $f(d(e_{mn}))$ is maximized if;

1. The costs to reach v_d from any node in P_i is optimal and decreasing since P_i is an optimal cost path.
2. Path P_i is the optimal delay path, $D(P_i) = \min_{P_i \in \Omega} D(P_i)$,

therefore the predictor multiplier w :

- a. Has the smallest delay from Ω , $\exists d(e_{mn}) \in \Omega$:
 $d(e_{mn})_{\min} = w$ or;
- b. Has the smallest delay statistics (e.g. mean link delays of P_i).

Property (2)

If all link costs have unit value and are equal, $\forall e_{mn} \in E$: $c(e_{mn}) = 1$, then $C_{jk(d)} = H_{jk(d)}$. The LCP is transformed into the Shortest Hop Path (SHP). The quantity $H_{jk(d)}$ is the optimum hop count to reach destination v_d from any node v_j via link e_{jk} (or adjacent neighbor node v_k). Equation (9) presents a problem for delay-constrained least-hop that is a special case of the DCLC problem. Equation (9) is the original TD routing inequality of DMFA [7]. In this work, we propose a new distributed algorithm for the QoS-routing problem of delay-constrained minimum-hop.

$$f(d(e_{mn})) = \left(\Delta - \sum_{e_{mn} \in P_{sj}} d(e_{mn}) \right) - (H_{jk(d)} \times w) \geq 0 \quad (9)$$

Property (3)

Under optimal conditions defined in **property (1)**, (9), which is a by-product of (6) is piecewise linear and monotonic decreasing with gradient $\nabla = H_{jk(d)}$. **Property (3)** holds when only all link delays are constant. Assume $\forall e_{mn} \in P_i$: $d(e_{mn}) = d$.

$$\lim_{d(e_{mn}) \rightarrow d} = \frac{f(d(e_{mn})) - f(d)}{d(e_{mn}) - d}$$

$$\lim_{d(e_{mn}) \rightarrow d} = \frac{[(\Delta - d(e_{mn})) - (H_{jk(d)} \times d(e_{mn}))] - [(\Delta - d) - (H_{jk(d)} \times d)]}{d(e_{mn}) - d}$$

$$\lim_{d(e_{mn}) \rightarrow d} = \frac{\Delta - d(e_{mn}) - (H_{jk(d)} d(e_{mn})) - \Delta + d + (H_{jk(d)} d)}{d(e_{mn}) - d}$$

$$\lim_{d(e_{mn}) \rightarrow d} = \frac{-(H_{jk(d)} d(e_{mn})) + (H_{jk(d)} d)}{d(e_{mn}) - d}$$

$$\lim_{d(e_{mn}) \rightarrow d} = \frac{-H_{jk(d)} (d(e_{mn}) - d)}{d(e_{mn}) - d}$$

$$\therefore \nabla f(d(e_{mn})) = -H_{jk(d)}$$

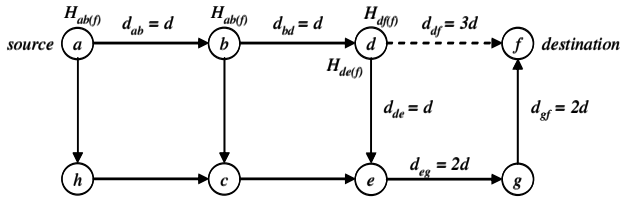


Figure 1. A directed topology

A. Monotonic Characteristics of the Inequality Function

It is imperative that a monotonic analysis is given as computation of DCMH is dependent on the monotonic characteristics of (9). Before proceeding with the analysis, we assume **property (2)** holds. Under optimal condition 1 and 2 as defined in **property (1)**, (9) is always monotonic decreasing. Decomposition of (9) produces two linear independent and piece-wise sub-functions with asymmetric monotonic characteristics.

- A. That $f_p(d(e_{mn})) = (H_{jk(d)} \times w)$ and exhibits either monotonic decreasing or increasing behavior.
- B. That $f_r(d(e_{mn})) = r_{j(d)}$ is always a strict monotonic decreasing function.

To guarantee clarity, a simple routing topology in Fig. 1 is considered with routing paths $P_1, P_2 \in P_{af}$, $P_1 = (e_{ab}, e_{bd}, e_{df})$, $P_2 = (e_{ab}, e_{bd}, e_{de}, e_{eg}, e_{gf})$.

Analysis 1: Property (1) conditions 1 and 2 are valid. The path P_1 is an optimal delay and optimal cost path.

Claim A:

$$f_p(d(e_{ab})) > f_p(d(e_{bd})) > f_p(d(e_{df}))$$

$$3d > 2d > d$$

Claim B:

$$f_r(d(e_{ab})) > f_r(d(e_{bd})) > f_r(d(e_{df}))$$

$$\Delta - d > \Delta - 2d > \Delta - 5d$$

Both sub-functions are monotonically decreasing functions. A delay-constrained hop optimized routing path is feasibly found when all conditions of **property (1)** are obeyed. Nevertheless, due to inexistence of the global-state for the proposed DCMH, it is very unlikely that condition 2 in **property (1)** is encountered during any routing process.

Analysis 2: Property (1) condition 1 does not hold when a diversion is required from the optimal path P_1 to non-optimal path P_2 . For example, a broken network link is detected or if $f_p(d(e_{mn}))$ is higher than $f_r(d(e_{mn}))$ at link e_{df} . Therefore, $C(P_2) \neq \min_{P_2 \in P_{af}} C(P_2)$ where path P_2 is non-optimal. At node d where the diversion is needed, $f_p(d(e_{mn}))$ reverses its order.

Claim A:

$$f_p(d(e_{ab})) > f_p(d(e_{bd})) < \overset{\text{increase}}{f_p(d(e_{de}))} > f_p(d(e_{eg})) > f_p(d(e_{gf}))$$

$$3d > 2d < 3d > 2d > d$$

Claim B:

$$f_r(d(e_{ab})) > f_r(d(e_{bd})) > f_r(d(e_{de})) > f_r(d(e_{eg})) > f_r(d(e_{gf}))$$

$$\Delta - d > \Delta - 2d > \Delta - 3d > \Delta - 5d > \Delta - 7d$$

Finding a delay constrained optimal hop path is unviable because of the increased routing costs at the diversion point. However, it is feasible to find a routing path that satisfies the delay constraint with minimum hop. It is worth mentioning that a diversion can also occur when $f_p(d(e_{mn}))$ reverses its order when its value is above $f_r(d(e_{mn}))$. The incurred higher cost due to routing path diversion does not change the monotonic decreasing order of $f_r(d(e_{mn}))$.

Analysis 3: Property (1) condition 2(a) and 2(b) are unreachable. Path P_1 is not an optimal delay path but a delay-constrained path. There is no prior knowledge that P_1 is the optimal delay path and w is the smallest in P_{af} . Another drawback with condition 2 is w varies with dynamics of network traffic. Assume d_{df} is doubled to $6d$.

Claim A:

Condition 2(a): $w = d(e_{mn})_{\min} = d$

$$f_p(d(e_{ab})) > f_p(d(e_{bd})) > f_p(d(e_{df}))$$

$$3d > 2d > d$$

Condition 2(b): $w_{i(av)} = \sum d(e_{mn})/h_i$, hence $w_{a(av)} = d$, $w_{b(av)} = d$ and $w_{d(av)} = 2.67d$

$$f_p(d(e_{ab})) > f_p(d(e_{bd})) < \overset{\text{Increase}}{f_p(d(e_{df}))}$$

$$3d > 2d < 2.67d$$

Claim B:

$$f_r(d(e_{ab})) > f_r(d(e_{bd})) > f_r(d(e_{df}))$$

$$\Delta - d > \Delta - 2d > \Delta - 8d$$

Variations in link delays has no effect on the order of $f_r(d(e_{mn}))$. It is always strictly monotonic decreasing. Sub-function $f_p(d(e_{mn}))$ exhibits an order decreasing and increasing with minimum and mean delay value of P_1 . A routing path that satisfies the delay-constraint with optimal hop is still achievable provided $f_p(d(e_{mn}))$ is maintained lower than $f_r(d(e_{mn}))$ value at each computation point. Finding the non-trivial value of predictor multiplier w guarantees a solution to $f_p(d(e_{mn}))$, and therefore (9).

The TD inequality (9) is satisfied when the predictor is lower than the residual-QoS. This is tantamount to finding a suitable w as defined in the non-trivial condition 2 of **Property (1)**. Nevertheless, satisfying **Property (1)** condition 2 or using any minimum value to represent w

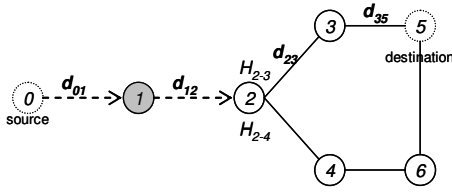


Figure 2. A simple network for path feasibility prediction analysis

only guarantee routing computation success but would not limit overheads. There is an inherent ‘weak’ satisfying capability of the inequality at initial computation phases since the predictor product would be considerably lower than residual-QoS (i.e. $r_{jk(d)} \gg H_{jk(d)} \times w$). Links with large delays have higher tendency of being selected resulting in an early “overspend” of the delay-constraint (or rapid draining of $r_{jk(d)}$). Consequently, routing beyond suffers multiple diversions due to frequent failures that are wasteful of message overheads and hop counts. In addition, the dual monotonic behavior of the predictor coupled with wide variance of link delays due to network traffic dynamics could also trigger diversions. One solution to limit the weaknesses of (9) and its vulnerabilities to network dynamics is to use w drawn from the constraints (hop count and delay) that is adaptive to changing routing behavior and network traffic.

B. Analytical Model of the Path Feasibility Predictor

Equation (9) is further extended in retrospect to its limitations as discussed in section A. Two different variables are defined further, D_{Fix} as denoted by (10) and D_1 . D_{Fix} is the ‘a priori’ mean link delay distributed equally over the links on the optimal path (shortest path) from any routing node v_j to destination v_d . It is not the actual delay, rather D_{Fix} gives an ‘artificial’ projection of the maximum allowable mean path delay if v_d is to be reached via the optimal hop path (i.e. $H_{jk(d)} + h$). Δ is the delay constraint and h is the traversed hop count to reach v_j on the tentative routing path from source v_s . This D_{Fix} is a varying quantity computed ‘online’ at each node. D_1 is an initial value also determined from (10) only once at v_s where $h = 0$ when $v_j = v_s$

$$D_{Fix} = \frac{\Delta}{H_{jk(d)} + h} \quad (10)$$

$$\left\{ \begin{array}{l} D_{Fix} = \frac{\Delta}{H_{jk(d)} + h} \rightarrow h \neq 0 \therefore v_j \neq v_s \\ D_1 = \frac{\Delta}{H_{sk(d)}} \rightarrow h = 0 \therefore v_j = v_s \end{array} \right.$$

Assume in Fig. 2 that a routing path exists as $P'_{sd} = (v_0, v_1, v_2, v_3, v_5; e_{01}, e_{12}, e_{23}, e_{35})$ where source is v_0 and destination is v_5 . At each node in P'_{sd} , the ‘true’ mean link delay is calculated and denoted as $D_{T,v}$ in (11). More specifically, $D_{T,v}$ is the tentative mean link delay from v_s

to any selected adjacent neighbor node v_v of current routing node v_j .

$$D_{T,v} = \frac{\left[\sum_{e_{mn} \in P'_{sj}} d(e_{mn}) \right] + d(e_{jv})}{h+1} \quad (11)$$

Example of $D_{T,v}$ sequence in Fig. 2

$$\left(\frac{d_{01}}{1} \right) \rightarrow \left(\frac{d_{01} + d_{12}}{2} \right) \rightarrow \dots$$

If link delays in P'_{sd} are equiprobable (i.e. $d(e_{01}) = d(e_{12}) = d(e_{23}) = d(e_{35}) = d$), then $D_{T,v} = D_{Fix}$. When this condition occurs, P'_{sd} is a delay-constrained path (i.e. accumulated delay always satisfies Δ). If the link delays are not equal, which is normal for network links due to dynamics of network traffic, a ‘mismatch’ exists between $D_{T,v}$ and D_{Fix} (i.e. $D_{T,v} \neq D_{Fix}$). Due to this ‘mismatch’, there is no guarantee that accumulated delay on routing path P'_{sd} would satisfy the Δ . This mismatch is represented as a ratio ϕ and is expressed as

$$\phi = \frac{D_{Fix} - D_{T,v}}{D_{Fix}} \quad (12)$$

The ϕ , when is subtracted from unity (one) yields a factor k (13). For the equiprobable case as is P'_{sd} , $k = 1$ because $\phi = 0$. The algorithm will potentially traverse the optimal hop path from v_s or any node v_j to v_d via any outgoing node v_k (i.e. $H_{sk(d)}$ or $H_{jk(d)}$). When $\phi \neq R^+$, $k > 1$, a path ‘penalty’ condition arises, and when $\phi = R^+$, $k < 1$, path ‘gain’ condition arises.

$$k = (1 - \phi) \quad (13)$$

$$k = \left\{ \begin{array}{l} (1 - \phi) = 1 \rightarrow D_{T,v} = D_{Fix} \rightarrow \text{equiprobable} \\ (1 - \phi) > 1 \rightarrow D_{T,v} > D_{Fix} \rightarrow \text{penalty} \\ (1 - \phi) < 1 \rightarrow D_{T,v} < D_{Fix} \rightarrow \text{gain} \end{array} \right.$$

The factor k is inserted into (9) as $(1-\phi)$ yielding a left-hand path predictor as denoted in (14) that is adaptive to varying link delays. This k factor is uniquely referred as the path feasibility predictor factor.

$$H_{jk(d)} \times w \times (1 - \phi) \leq \Delta - \sum_{e_{mn} \in P'_{sj}} d(e_{mn}) \quad (14)$$

The predictor multiplier w is replaced with D_1 in (14) to give a final routing inequality of Eq. 5.15 as such that consistency is reached for (10) (i.e. $D_{T,v} = D_{Fix}$ when $h = 0$, indicating that the delay-constrained path is an optimal hop path). Equation (15) is termed as the Path Feasibility Prediction (PFP) inequality for differentiation with the TD inequality (9).

$$H_{jk(d)} \times w \times (1 - \phi) \leq \Delta - \sum_{e_{mn} \in P'_{sj}} d(e_{mn}) \quad (15)$$

There are two primary strengths of the PFP inequality over the TD inequality. First, the inclusion of the ‘a-priori’ feasibility predictor factor $k = (1-\varphi)$ transforms the TD inequality (9) from a blind computation model into a predictive based computation model (15). Second, substituting w with D_1 stems from the inherent weak satisfying criteria of the TD inequality. In addition, the casual or random characteristic of the TD inequality is reduced where selection of links on a routing path is logically influenced by the initial mean D_1 , derived from an application’s hop count and delay-constraint requirement.

One potential use of D_1 in (15) is found in routing of hard real-time communication channels [23]. Without making specific linkage to either rate-based or scheduler-based scheduling policies [4], D_1 can be mapped as the target deadline or worst-case delay that can be offered to a connection on each link in the routing path P'_{sd} . Link deadlines [6] can be calculated by knowing: the source and destination endpoints (assuming hop counts on the optimal path); the traffic model consisting of minimum inter-arrival time, maximum packet length and maximum burst size; the end-to-end delay bound. The calculation for the link deadlines is inherently influenced by the required resources of a connection. They are link bandwidth and buffer space requirements. By substituting link datelines in (15) as D_1 , routing would aid jitter control [24], thus potentially minimizing some buffer space requirements by computing a path with a low link to link delay variations.

C. Path Gain and Path Penalty Property of the PFP Model

We elaborate further in this section the adaptability and influence of the path feasibility factor k to variations in link delays and route corrective capability for the proposed DCMH. Two mismatched cases are considered in Fig. 5.2 for link d_{01} where $d_{01} > d_{12}$ and $d_{01} < d_{12}$.

Case 1. $d_{01} > d_{12}$; assume $d_{01} = d$ and $d_{12} = 0.5d$

$$D_{T,1} = \frac{d_{01} + d_{12}}{2} = \frac{d + 0.5d}{2} = 0.75d$$

$$\varphi = \frac{D_{Fix} - D_{T,1}}{D_{Fix}} = \frac{d - 0.75d}{d} = 0.25$$

$$k = (1 - 0.25) = 0.75$$

Case 2. $d_{01} < d_{12}$; assume $d_{01} = d$ and $d_{12} = 2d$

$$D_{T,1} = \frac{d_{01} + d_{12}}{2} = \frac{d + 2d}{2} = 1.5d$$

$$\varphi = \frac{D_{Fix} - D_{T,1}}{D_{Fix}} = \frac{d - 1.5d}{d} = -0.5$$

$$k = (1 - (-0.5)) = 1.5$$

As delay on link d_{12} delay is increased to $2d$, the ‘exact’ mean $D_{T,1}$ rises above the maximal allowable fixed mean D_{Fix} resulting in the mismatch factor φ to have a negative value. The net value of the predictor is hence higher because of the up-scaling of the k factor from 1.0 to 1.5. For case $d_{01} > d_{12}$, when $d_{12} = 0.5d$, $k = 0.75$. The net predictor product is down-valued by a factor of 0.75 (or 75%) yielding a path gain of 0.25. Subsequently, if there is an increase in the delay of the next link by about 25% of the D_{Fix} (i.e. $d_{23} = 1.25d$), the PFP inequality (15) will still hold. Another interesting aspect of the k factor is adaptability to routing path diversions. A diversion causes a routing computation to veer from an optimal to a non-optimal route resulting in a fine-grained tuning of D_{Fix} . Beyond this diversion point, routing computation must find a path with a tentative mean $D_{T,v}$ lower or equal to that new lower tuned D_{Fix} . The variable D_{Fix} has an inverse proportional relationship with the quantity $H_{jk(d)} + h$. Equation (16) shows the cost offset denoted as h_{div} due to routing diversions at any routing computation point. $H_{sk(d)}$ is the optimal hop cost from source s to destination d via outgoing node v_k .

$$h_{div} = H_{jk(d)} - H_{sk(d)} \tag{16}$$

As more routing diversions are encountered, h_{div} increases making D_{Fix} smaller. The mismatch ratio φ is more negatively biased therefore inflating the left-hand predictor product. This condition is however an advantage since it could prevent routing computation from being erratic saving precious computational messages. Any slightest variation between D_{Fix} and $D_{T,v}$ either due to large fluctuations in network link delays or routing path diversions is captured by φ that ultimately affects the predictor. The desired objective is to always maintain $k \leq 1$ (unity) on the tentative routing path as the algorithm traverses the path. It is important to mention that a diversion can only happen due to a mismatch between D_{Fix} and $D_{T,v}$ so significant that the PFP inequality is violated. The k factor is inferred to be effective for finding a balanced distribution of network link delays on any optimal routing path, and to limit or prevent routing computation from going astray due to diversions on the non-optimal path.

IV. ROUTING INFORMATION AND TABLE STRUCTURE

In this section we discuss routing information that must be available for the execution of DCMH. Each node owns a routing table containing routing information for the computation of the PFP inequality and on the delay-constrained connection session. Assuming many sessions exists with distinct session identifier S_{id} . The entry in the routing table for a S_{id} at any node i is shown in Table 1.

TABLE I. ROUTING TABLE INFORMATION

$H(S_{id}, i, j)$	SP hop count to reach the destination via each adjacent neighbor node j from node i
$LINK_DLY(i, j)$	The updated delay of links connecting node i to each adjacent neighbor node j
$PATH_DLY(S_{id})$	The accumulated path delay to reach node i from the source

A global-state database that represents the topological connection (binary connectivity) of the network must be kept. Each node uses this ‘global connectivity map’ to compute the SP hop counts to all other nodes (all possible destinations) using Dijkstra’s [17] algorithm. It sorts the $H(S_{id}, i, j)$ entry giving precedence to adjacent neighbor nodes with the fewest hop counts to reach the destination e.g. $H(S_{id}, i, m) < H(S_{id}, i, n) < H(S_{id}, i, k)$ where m, n, k are direct neighbors of i . This means that adjacent neighbors with the fewest hop counts are always given priority for evaluation. We refer to the specification of OSPF [10] that explains the procedure of state update and maintenance of the global-state database. Nevertheless, the only global information that DCMH needs to store is binary topological information (i.e. for calculation of the SP hop counts) and, addition and failure of nodes and links that changes the network topology is expected to be infrequent. The SP hop counts can be pre-computed and kept in a ‘global SP hop count table’ provided network topology is intact. The storage cost and protocol overhead for maintaining this global topological database is minute compared with algorithms that depend on global-state of dynamic link metrics (e.g. delay). The proposed DCMH is oblivious to the problem of state information imprecision as this is particularly related to the staleness of global-state [22], [20]. Routing table explosion is instantaneous and efficient where the size is a function of the number of ongoing connection sessions. A connection request is sequentially passed from one node to another if only the ‘a priori’ condition of the PFP inequality is met. This implies that the routing table is not a forwarding table. Based on routing information requirements of Table 1 and the PFP inequality (15), DCMH is considered as hybrid, where route selection is based on a pre-computation (i.e. $H(S_{id}, i, j)$) and on an on-demand (i.e. $LINK_DLY(i, j)$) approach.

V. ROUTING CONTROL MESSAGES

In distributed routing algorithms, routing path computation and setup are done either by exchanging a control message in a sequential mode or many control messages in parallel. These control messages are ‘made-to-measure’ whose size and information is dependent on the requirements of an algorithm. The standard criteria in defining a control message is by determining how much information needs to be communicated, and communication bandwidths and nodal buffer spaces should not be overly consumed. For example, in [18] the control message size is limited to the size of an ATM cell for seamless and efficient message processing in the routers. For many prominent DRAs [14], [18], a standard ‘fail-safe’ approach is to record routing path information at intermediate nodes so that small control message size is utilized. An opposite strategy is adopted for DCMH using bigger message space primarily for routing loop detection. This implies that no mechanism for routing loop removal is needed for DCMH.

Three control messages are defined to implement DCMH. They are **PATHRQST**, **CRANKBACK** and **PATHACK** message. **PATHRQST** is a connection request passed from the source to the destination sequentially traversing any intermediate nodes to build the routing path. **CRANKBACK** is passed in reverse to the predecessor node from the node where the routing instance is currently at on the tentative routing path. Assuming that routing instance is at v_2 , as in Fig. 2, the successor and predecessor of v_2 is v_3 and v_1 respectively. Node v_2 can either forward a **PATHRQST** to v_3 or return a **CRANKBACK** to v_1 depending on the outcome of tentative path delay tests. When a delay-constrained routing path is found, **PATHACK** is sent in reverse along the successfully computed routing path to reserve the resources and admit the connection.

- **PATHRQST**($S_{id}, src, r\{ \}, \Delta, D_1, acmDLY, h$)
- **CRANKBACK**($S_{id}, src, r\{ \}, \Delta, D_1, h$)
- **PATHACK**(S_{id}, src)

Here, src is the source address, dst is the destination address, $r\{ \}$ is the list of nodal address that formed the partial or complete routing path, Δ is the delay constraint, D_1 is the initial delay, $acmDLY$ is the accumulated delay from the source and h is the distance in hops the message has traveled from src . The control messages are only active during the path computation phase. For any connection session S_{id} , the control messages are discarded once a path is found (when **PATHACK** message reaches src) or when no feasible path exists (when **CRANKBACK** is returned to src). The control messages can only travel a maximum distance of one hop at any computation instance.

VI. DELAY-CONSTRAINED MINIMUM-HOP (DCMH) DISTRIBUTED ROUTING ALGORITHM

In this section, the heuristics of the proposed DCMH algorithm is explained. Upon receipt of a connection establishment request to a destination dst , with a delay constraint Δ . A source src , initiates DCMH by issuing a unique connection session identifier S_{id} . The src builds the routing table, initialize $h = 0$ and $acmDLY = 0$, and is now the active routing node referred to as the ‘head’. Any node receiving a **PATHRQST** or a **CRANKBACK** also assumes the role as *head*. The *head* selects an adjacent neighbor node denoted as successor ‘*succ*’ from $H(S_{id}, i, j)$ table entry ($head = i$ and $succ = j$) and calculates the tentative path delay denoted in (I). The $tent_dly(src, succ)$ is the tentative path delay up to the selected *succ*.

$$tent_dly(src, succ) = acmDLY(S_{id}, head) + LINK_DLY(head, succ) \quad (I)$$

$LINK_DLY(head, succ)$ is the local link delay connecting *head* and *succ*. The $acmDLY(S_{id}, head)$ is the accumulated path delay from src to *head* extracted from the **PATHRQST** received by the *head*. The *head* evaluates.

$$tent_dly(src, succ) \leq \Delta \quad (II)$$

If (II) is satisfied, the *head* evaluates (III). It tells the *head* how many hops it is located away from *dst* via the *succ*.

$$H(S_{ID}, head, succ) = 1 \quad (III)$$

When (III) is satisfied, the *head* is one final hop from *dst* where *succ* = *dst*. The *head* sends a final **PATHRQST** to *dst*. The delay-constrained routing path is found. Destination *dst* returns **PATHACK** towards the routing path in reverse to confirm the connection thereby reserving resources.

$$H(S_{ID}, head, succ) > 1 \quad (IV)$$

If otherwise (IV) is reached, routing must traverse more than one hop to reach *dst*. It indicates that *succ* ≠ *dst*. This condition calls the *head* to evaluate the PFP inequality (Eq. 5.15). Using (I) and *h* from the received **PATHRQST**, the *head* calculates D_{Fix} (Eq. 5.10), $D_{T,v}$ (Eq. 5.11) and ϕ (Eq. 5.12) to evaluate the PFP inequality. If the PFP inequality is satisfied, the *head* stores $acmDLY(S_{ID}, head)$ in $PATH_DLY(S_{ID})$ table entry. Before forwarding **PATHRQST** to the *succ*, the *head* updates **PATHRQST** contents (i.e. $acmDLY(S_{ID}, head) = tent_dly(head, succ)$), increments *h* by 1 and appends its address in $r\{ \}$. The *succ* becomes a new *head* subsequent to accepting the **PATHRQST**.

When either PFP inequality or (II) is violated, the *head* selects another successor from $H(S_{ID}, i, j)$ table entry. If all successors are selected and individual evaluation fails, routing computation at *head* terminates. The *head* sends a **CRANKBACK** to its predecessor. Predecessor address is listed in $r\{ \}$ contained in the **PATHRQST** sent earlier to *head*. The *head* omits this predecessor address from $r\{ \}$ and decrements *h* by 1 before sending the **CRANKBACK**. Routing table entry created earlier is hence deleted by the *head*. The predecessor becomes the *head* upon receiving the **CRANKBACK**. This re-appointed *head* utilizes the formerly stored $PATH_DLY(S_{ID})$ as $acmDLY(S_{ID}, head)$ and revisits the evaluation. DCMH routing computation is terminated if **CRANKBACK** is returned all the way to *src*. A *head* must select a *succ* according to its sorted list (i.e. precedence for *succ* with lowest SP hop count), a *head* cannot select a *succ* that has already been selected, a *head* must ensure that a selected *succ* is not listed in $r\{ \}$ to avoid routing loops. The distributed heuristics for DCMH is given below.

A. Routing Loops

Avoiding or elimination of routing loops is important for QoS-routing algorithms. Routing loops are prevalent in a DRA that depend on the global-state when the routing tables are outdated. Most DRAs generally experience ‘forwarding loops’ when a connection request sent by a node is either returned to itself or forwarded to any node in the set that already formed the partial routing path. A large majority of DRAs eliminate these frequently occurring forwarding loops using additional

```

1. BEGIN
2. get a succ from H(SID, i, j) entry
3. switch (CASE 1 or CASE 2)
4. CASE 1: if (∃ succ ∈ H(SID, i, j): succ unselected && succ
   ∉ r{ })
5.     if ( tent_dly(src, succ) > Δ )
6.         loop: go to line 2
           /*get another successor*/
7.     else if ( tent_dly(src, succ) ≤ Δ )
8.         if ( H(SID, head, succ) = 1 )
9.             send final PATHRQST to dst
10.        END ROUTING
           /*delay-constrained path is found to dst*/
11.    else if ( H(SID, head, succ) > 1 )
12.        if ( predictor ≤ residual-QoS )
13.            send PATHRQST to succ
           /*successor becomes head*/
14.        else if ( predictor > residual-QoS )
15.            loop: go to line 2
           /*get another successor*/
16. CASE 2: if (∀ succ ∈ H(SID, i, j): succ selected || succ ∈ r{ })
17.     if (head ≠ src)
18.         send CRANKBACK to predecessor
           /*predecessor becomes head*/
19.     go to line 2
20.     else if (head = src)
21.         END ROUTING
           /*connection establishment failed*/

```

loop removal mechanisms [14]. This loop elimination phase adds extra complexity and incurs more messages, in addition to being time consuming. One advantage is small size of request messages can be used. To avoid forwarding loops from occurring with DCMH, routing path nodal addresses are recorded in the **PATHRQST** and **CRANKBACK** message. In the DCMH heuristic, an identity examination is performed by the *head* to ensure that the selected successor *succ* is not enlisted in $r\{ \}$. This preventative technique avoids routing loops from ever occurring during routing computation. This scheme is fundamentally simple and efficient to implement. One potential drawback is **PATHRQST** message payload may increase linearly with hop distance. Sudden termination in routing computation is inevitable if the message payload is out-of-bounds of the message size. Nevertheless, two constructive and logical explanations are given to trivialize the perceived problem.

1. The length of a loop may only span a few nodes away. Only small size loops exist in most instances of forwarding loop occurrences. In addition, studies [15], [16] indicate that the Internet possess a ‘small-world’ property where the average hops-per-path ratio for many combinations of routing paths is small. For the

reasons above, we say that small size of $r\{ \}$ is needed in the **PATHRQST** and **CRANKBACK** message.

- If the IPv4 [10] address size (i.e. 32 bits) is considered, a 10 hop routing path only occupies a small 40 bytes message space to carry the full addresses of the routing path nodes $r\{ \}$. Router address size for IPv6 [11] is increased to 128 bits. The specifications of IPv6 demand the minimum link Maximum Transfer Unit (MTU) to be at 1280 bytes. Again, a 10 hop routing path consumes 160 bytes payload space of the allocated 1240 bytes data size (+ 40 bytes of header). Suffice to say, **PATHRQST** and **CRANKBACK** messages will not be lost or dropped because the control messages would not undergo packet fragmentation. The size of routing information contained in **PATHRQST** and **CRANKBACK** messages is guaranteed to not consume link bandwidth and buffer space.

B. Time and Computational Complexity

The time complexity of a distributed routing algorithm concerns the total time needed to establish a connection from a source to a destination. This corresponds to a single round trip time of the call connection phase. For sequential distributed computation of the DCMH, exactly one single message is spent on each link, equivalent to one unit time of message execution. If the length of the path from the source to the destination is given as L , then the bound for the time complexity of DCMH is of $O(2L)$ units of time.

Optimally, DCMH reaches the destination in the shortest hop distance H_{opt} where each node on the optimal routing path selects only the first successor listed in the routing table. Routing crankbacks are non-existent. There is at most one **PATHRQST** and **PATHACK** message on each link, round trip on the optimal hop path. The best-case computational complexity is linear and is of $O(2H_{opt})$. In the worst-case complexity, DCMH will have traversed all possible paths between the source and the destination. Each link on the enumerated paths will experience a crankback. The worst case computational effort required in enumerating all paths in a completely connected network is presented in [9] using a method called 'sliding-window'. For a network with N nodes and D degree, (17) represents the upper-bound on the computational complexity in finding all possible combinations of routing paths between any source-destination pair.

$$F(N, D) = (N^2 - 5N + 8)(D - 1) + 1 \quad (17)$$

For a completely connected network, $D = N - 1$; by neglecting the constants and removing the smaller order of the function, the worst case computational complexity is polynomial and bounded by $O(N^3)$. Remarkably, simulation shows that the average number of messages exchanged for DCMH is much lower than the worst case and DCMH's results are closely matched to the best-case complexity.

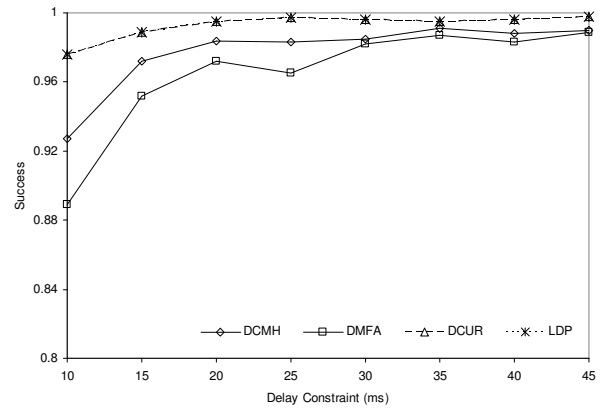


Figure 3. Connection success, network size of 100 nodes, delay-constraint from 10ms to 45ms

VII. SIMULATION AND RESULTS

We developed a routing application to simulate the routing performance of DCMH in random networks and compare against DMFA [7], Delay-Constrained Unicast Routing (DCUR) [14] and LDP (i.e. Dijkstra's) [17]. DCUR keeps a cost vector and a delay vector table, and alternates between the two to search for a suitable delay-constrained least-cost path. A random model [8] is used to generate a realistic random topology with an average node degree of four. The graphs were generated until a fully connected random network with a minimum individual node degree of one is found. Nodes were placed in a rectangular coordinate of 1000×1000 km². Network links were bidirectional with homogenous capacities of 155Mb/s. Link delays were asymmetric and were a function of propagation and queuing delay, $D(i, j) = p(i, j) + q(i, j)$ where $p(i, j) = p(j, i)$ and $q(i, j) \neq q(j, i)$, therefore $D(i, j) \neq D(j, i)$. A random amount of traffic was generated on each link denoted as $arrvRate$, and a source and destination were randomly selected and referred to as a connection request session. The link utilization factor, ρ was randomly generated between 0 and 0.8 with a uniform distribution. Queuing delay on each link was taken as, $\rho / (1 - \rho) \times (1 / arrvRate)$. In each session, the number of messages exchanged to construct a path was recorded. The session procedure was repeated until confidence intervals below than 5% of the average value was achieved, using 95% confidence levels. Simulation was repeated for varying network sizes from 20 to 200 nodes across 8-point delay constraints from 10ms to 45ms.

A. Performance Comparison with DMFA, DCUR and LDP algorithm

The connection success performance for 100-node network size is shown in Fig. 3. DCUR and LDP achieved optimum success performance since both algorithms are able to determine a feasible delay-constrained path from its global-state. We calculated the overall blocking percentage relative to LDP. Connection session blocking for DCMH is only 1.5% compared to

DMFA, which is 3.33%. Blocking for DMFA is higher for tight delay-constraints i.e. $\Delta_{10ms} = 10.2\%$. DMFA's poor connection success is due to two factors. First, the variable $d(e_{mn})_{min}$ in (9) that is a natural variable of link delays in the network can be high if the routes which DMFA follows lies on the network or sub-network that is loaded with high traffic. For loaded networks with high hops-per-path ratio, the multiplicative product of $H_{jk(d)} \times d(e_{mn})_{min}$ in (9) will be huge. As a result, routing computation in DMFA is unable to traverse some potential paths. Second, when either *tentative_delay* > Δ or TD (9) is violated, DMFA initiates a crankback. This redundancy limits the chances of finding a feasible routing path simply because of the increase in the predictor product, contributed by a higher hop count to destination quantity $H_{jk(d)}$, for every crankback suffered.

Despite being slightly better than DMFA, success rates for DCMH are lower compared to DCUR and LDP. We infer that the fixed mean, D_{Fix} (10) in (15) contributed to the drop in connection success. For relatively strict (small) Δ and increasing hop size (a routing path with high hop length), the numerator in (10) causes D_{Fix} to be small. If the tentative mean, $D_{T,v}$ (11) of the computed path is close or above D_{Fix} , this condition will consequently cause the mismatch ratio, ϕ (12) to have negative values. As a result, the factor $k \geq 1$, causing the predictor to be penalized more often. The total product $H_{jk(d)} \times D_1 \times (1-\phi)$ becomes tight and is restrictive of forward movement. DCMH was forced to make frequent crankbacks and was therefore not able to explore many paths. Unlike DMFA, optimum connection success is achievable by 'softening' of the initial mean, D_1 (10) which would compensate the predictor if the path followed is restrictive. The D_1 has an upper-bound value when its denominator is the SP hop count between the source and the destination, (i.e. when $H_{jk(d)} = H_{sk(d)}$ for $v_j = v_s$).

$$NOF_x = \frac{Delay_x - Delay_{LDP}}{Delay_{LDP}} \quad (18)$$

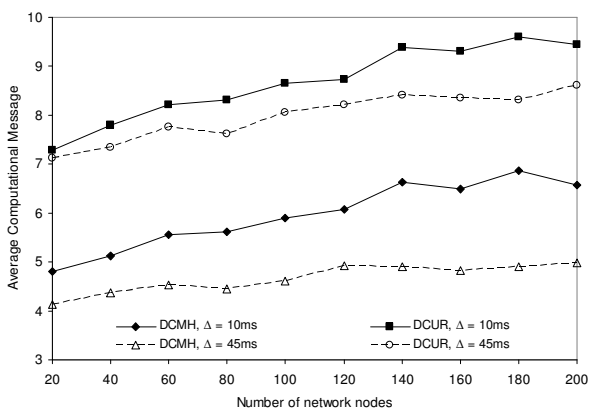


Figure 4. Average message overhead, network size from 20 nodes to 200 nodes, delay-constraint of 10ms and 45ms

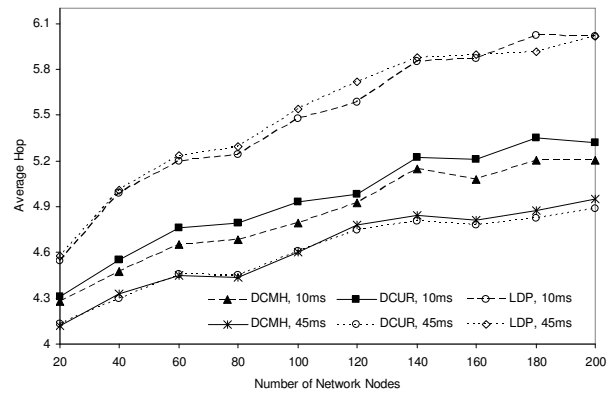


Figure 5. Average hop distance traversed, network size from 20 nodes to 200 nodes, delay-constraint of 10ms and 45ms

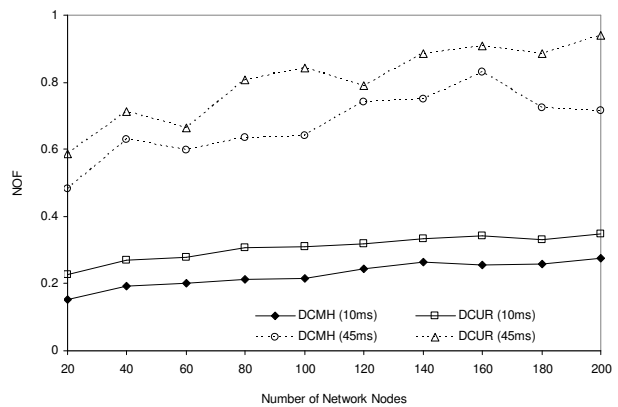


Figure 6. The NOF of DCMH and DCUR relative to LDP at delay constraint of 10ms and 45ms

In most computation session, the successfully computed routing path would not be optimal because it would deviate by one or more hops away from the optimal length i.e. $H_{jk(d)} = H_{sk(d)}$. This deviated hop value is actually the h variable found in (10). Knowing this h value 'a priori' and utilizing it in setting an appropriate value for D_1 i.e. $D_1 = H_{jk(d)} + h$, can guarantee higher connection success performance and even better path predictability.

Figure 4 shows the average message overhead versus varying network sizes for DCMH and DCUR. Average message growth is slow and logarithmic in the increasing network size. The computational overhead of DCMH is lower than DCUR. This is because DCUR spent three different control messages as the distributed heuristics alternates between the least-cost and least-delay paths at each routing node. DCMH on average exchanged about three fewer message compared to DCUR to construct delay-constrained paths at $\Delta = 45ms$, e.g. $N = 100$, DCMH = 4.6 and DCUR = 8.0 messages, $N = 200$, DCMH = 5.0 and DCUR = 8.6 messages.

The average number of hops traversed for successful connections is shown in Fig. 5. We first say that for LDP that uses Dijkstra's algorithm to compute shortest-delay path to the destination, this delay-optimization is achieved at the expense of high hop cost. For relatively strict Δ , only a limited set of routes are available. The

probability of available routes with short hop lengths that satisfies the Δ is low. In addition, the strict Δ causes D_{Fix} to be small resulting in the selection of links with low delays on the path. For $\Delta = 10\text{ms}$, the hop cost of DCMH is moderately better compared to DCUR. As the Δ is increased to 45ms , a route with smaller hop distance that satisfies the Δ is more preferred than a route that optimizes the delay. This suggests that links with higher delays are more likely to be part of the computed path. The quantity, $H_{jk(d)}$ in (15) means that DCMH prioritizes and so chooses minimum hop routes instead of shortest delay routes. Depending on the Δ level and network size (or the hop count between a source-destination pair) that ultimately determines D_{Fix} , DCMH inherently can behave from an algorithm that prioritizes delay to more like one that optimizes hop cost. From Fig. 5, the average measured cost at $\Delta = 45\text{ms}$ for DCMH is comparable to of DCUR. DCMH is able to achieve good connection success rate using modest message overhead with minimal hop-cost, and at the same time satisfying the delay-constraint without ever need to store additional global-state information. One limitation of DCMH as discussed is the restrictive computation of a feasible delay-constrained path when the predictor product of the PFP becomes tight. Nevertheless, we have provided a logical explanation on negating connection session blocking of DCMH.

B. Analysis of Distribution of Delays over Routed Paths

One of the important QoS requirements of delay-sensitive multimedia applications is maintaining low jitter. Conventional methods of controlling jitter are to control the rate of packet streams by sources and provide adequate buffer spaces along the computed path to avoid packet loss. Because jitter is partly due to variations in queuing delays and propagations, we believe QoS-routing algorithms can aid jitter control [24] by computing a routing path that has a low variance of delays (i.e. low link to link path delay variations). In this simulation, we measured path delay variances of DCMH and DCUR in random topology for the network size of 100-nodes. From the recorded results, we analyzed the 100-sample (i.e. $\alpha = 0.02$) long term profile of path delay variances using an Exponential Weighted Moving Average (EWMA). A similar simulation procedure was used to generate random topology and network link delays for this simulation. We measured path delay variances of DCMH and DCUR when both equally retrieved a delay-constrained solution with a hop cost of 6 hops.

Figure 7 shows the measured path delay variances and the moving average (i.e. EWMA) of DCMH and DCUR for $\Delta = 20\text{ms}$. We infer two opinions. First, paths computed by DCMH are more likely on average to have smaller computed path delays. We demonstrate this opinion by referring to the Non-Optimality Factor (NOF) plot in Fig. 6 where results indicate better optimality of DCMH than DCUR. The NOF (18) represents a measure of inefficiency of a distributed routing algorithm x (i.e. DCMH and DCUR) compared to the optimal path computed by LDP. Second, paths computed by DCMH would present lower delay variance compared to DCUR.

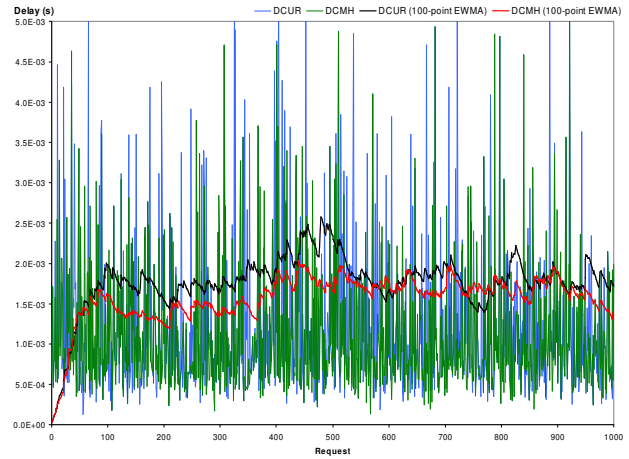


Figure 7. The EWMA of computed paths delay variance at $\Delta = 20\text{ms}$

In Fig. 7, DCMH exhibits smaller long term average fluctuations of variances compared to DCUR. Results in Fig. 7 most importantly indicate that D_1 and D_{Fix} hence the path feasibility predictor $k = (1-\phi)$ (13) prevents (or limits) a computed path from having a wide path delay variance. This is an inherent feature of the PFP inequality of DCMH.

VIII. CONCLUSION

In this paper, we presented a new distributed delay-constrained routing algorithm. Our DCMH is unique from other DRAs as it employs a novel adaptive path feasibility predictor that is able to make an ‘a priori’ prediction of the feasibility of a tentative routing path, done intelligently by utilizing both global topological and local link-state information. We evaluated through simulation and compared DCMH against the well-known DCUR and LDP. Average message performance of DCMH is modest and better than DCUR for the same measured cost performance. DCMH toggles from being a delay-optimization to a delay-constrained hop-minimization algorithm. The proposed DCMH is very simple to implement, highly scalable, efficient and provides greater insensitivity to the staleness of the link-state than algorithms that rely on global link-state information such as DCUR and LDP. Also, nodal storage requirements are minimal in comparison with many contemporary routing algorithms. Most importantly, we have demonstrated with DCMH and the PFP model that for a DRA that employs the local-state where no global information is present, suitable computation models can be devised that are adaptive and intelligent to improve routing performance. This strategy can best achieve overall scalability of QoS-routing algorithms. We believe that DRAs are equally useful and important as SRAs for some routing problems where the existence of the global-state is trivial.

IX. REFERENCES

- [1] S. Chen and K. Nahrstedt, "An Overview of Quality of Service Routing for Next Generation High-Speed Networks", *IEEE Networks Mag.*, November/December, 1998, pp. 64-79.
- [2] A. R. M. Shariff and M. E. Woodward, "A Distributed Algorithm for Unicast QoS-Routing using Path Feasibility Prediction", *IEEE Workshop on High Perf. Switching and Routing*, Poznan, June 2006.
- [3] G. Apostolopoulos, R. Guerin, S. Kamat and S. K. Tripathi, "Quality of Service Routing: A Performance Perspective", *ACG SIGCOMM Comp. Comm. Rev.*, vol. 28, issue 4, October, 1998, pp. 17-28.
- [4] H. Zhang and S. Keshav, "Comparison of Rate-Based Service Disciplines", *ACM SIGCOMM Comp. Comm. Review*, vol. 21, pp. 113-121, September 1991.
- [5] S. Lin and D. J. Costello, "Error Control Coding: Fundamentals and Applications", *Prentice Hall*, Englewood Cliffs, New Jersey, 1983.
- [6] D. D. Kandlur, K. G. Shin and D. Ferrari, "Real-Time Communication in Multi-Hop Networks", *11th Intl. Conf. on Distributed Computing Sys.*, pp. 300-307, 1991.
- [7] S. S. Deb and M. E. Woodward, "A New Distributed QoS Routing Algorithm based on Fano's Method", *Computer Networks Journal*, vol. 48, 2005, pp. 155-174.
- [8] M. Doar and I. M. Leslie, "How Bad is Naïve Multicast Routing", *IEEE INFOCOM*, San Francisco, USA, 1993.
- [9] M. E. Woodward and M. M. M. Al-Fawaz, "Partitioning Techniques for Reducing Computational Effort of Routing in Large Networks", *IEE Proc., Special Issue on Superhighways Technology and Broadband VPN*, vol. 151, pp. 95-100, February 2004.
- [10] The IETF, "RFC 791 – DARPA Internet Protocol Specification", September 1991.
- [11] The IETF, "RFC 2460 – Internet Protocol, Version 6 (IPv6) Specification", December 1998.
- [12] R. M. Fano, "A Heuristic Discussion of Probabilistic Decoding", *IEEE Trans. on Information Theory*, vol. 64, no. 9, April 1963.
- [13] J. M. Wozencraft and I. M. Jacobs, "Principles of Communication Engineering", *Wiley*, New York, 1965.
- [14] D. S. Reeves and H. F. Salama, "A Distributed Algorithm for Delay-Constrained Unicast Routing", *IEEE/ACM Trans. on Networking*, vol. 8, no. 2, pp. 239-250, April 2000.
- [15] X. F. Wang and G. Chen, "Complex Networks: Small-World, Scale-Free and Beyond", *IEEE Circuits and Systems Magazine*, pp. 6-20, 1st Quarter, 2003.
- [16] C. Faloutsos, P. Faloutsos and M. Faloutsos, "On Power-Law Relationship of the Internet Topology", *Proc. of the ACM SIGCOMM*, September 1999.
- [17] T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms", *McGraw Hill*, 1991.
- [18] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing", *Proc. of the 23rd Annual IEEE Conf. on Local Computer Networks*, pp. 80, October 1998.
- [19] E. Crawley, R. Nair, B. Rajagopalan and H. Sandick, "A Framework for QoS-based Routing in the Internet", IETF RFC 2386, August 1998.
- [20] G. Apostolopoulos et al., "Intradomain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis", *IEEE Network, Special Issue on Integrated and Differentiated Services for the Internet*, vol. 13, no. 5, pp. 42-54, September/October 1999.
- [21] F. A. Kuipers and P. V. Mieghem, "The Impact of Correlated Link Weights on QoS Routing", *IEEE INFOCOM*, San Francisco, USA, 2003.
- [22] A. Shaikh, J. Rexford and K. G. Shin, "Evaluating the Impact of Stale Link State Quality-of-Service Routing", *IEEE/ACM Trans. on Networking*, vol. 9, no. 2, April 2001.
- [23] C. M. Aras, J. F. Kurose, D. S. Reeves and H. Schulzrinne, "Real-Time Communication in Packet Switched Networks", *Proc. of the IEEE*, vol. 82, pp. 122-139, January 1994.
- [24] D. C. Verma, H. Zhang and D. Ferrari, "Delay Jitter Control for Real-Time Communication in a Packet Switching Network", *IEEE Proc. of TRICOMM 91*, pp. 35-43, April 1991.

A.R. Mohd Shariff was born in Penang, Malaysia in 1974. He received a B.Eng(Hons) in Electrical and Electronics degree in 1998 from the University of Plymouth (UK). In 2002, he obtained his MSc. in Personal, Mobile and Satellite Communications from the University of Bradford (UK). He is currently with the Department of Computing at the same institution pursuing a Ph.D degree in Communication Networks.

He worked as a Design and Development Engineer at Motorola CGISS R&D (Penang) from 1999 until 2001 where he was responsible for the developments of UHF VCO and PLL circuits, receiver systems and antennas of two-way analog and TETRA digital portables. His current research interests include fixed and wireless ad-hoc routing, QoS and ubiquitous networking.

Mr. Mohd Shariff is a member of the IEEE society and a recipient of the Malaysian government scholarship.

M. E. Woodward was born in Derby, England in 1941. He graduated with a 1st class honours degree in Electronic and Electrical Engineering from the University of Nottingham (UK) in 1967 and received a Ph.D degree from the same institution in 1971 for research into decomposition of sequential logic systems.

In 1970, he joined the staff of the Department of Electronic and Electrical at Loughborough University (UK) as a lecturer, being promoted to Senior Lecturer in 1980 and Reader in Stochastic Modelling in 1995. He remained at Loughborough until 1998 when he was appointed to the Chair in Telecommunications at University of Bradford (UK) where he also became the Director of Telecommunications Research Centre. His current research interests include queuing networks, telecommunications traffic modeling, quality of service routing and mobile communications systems.

Professor M. E. Woodward is the author of two books and over one hundred research papers. He holds two ESPRC grants and is supervisor to fifteen full time research students. He is a Fellow of the Institute of Mathematics and its Applications (FIMA) and is both a Chartered Mathematician (CMath) and Chartered Engineer (CEng).