一种端到端网络的不相交多路径 QoS 路由算法*)

朱尚明[1] 庄新华[2] 高大启[1]

(华东理工大学计算机科学与工程系 上海 200237)[1] (美国密苏里大学计算机科学系 美国 MO65211)[2]

摘 要 不相交多路径路由算法旨在一个端到端的网络中为应用流选择多个路径,且这些路径在瓶颈链路上是彼此不相交的。本文提出的不相交多路径 QoS 路由(DMQR)算法在 Dijkstra 最短路径和最短最宽路径(SWP)算法的基础上,能够动态地计算时延最短、带宽最宽、在瓶颈链路上互不相交的路径,且保证每个路径都是满足一定服务质量的。在视频会议、远程医疗和远程教育等重要的视频通信场合,要求应用层和网络层必须协同工作以保证一些必要的 QoS,例如端到端的带宽、时延和包丢失率等。本文针对端到端的网络,重点讨论不相交多路径 QoS 路由算法在应用层的设计和实现。性能分析和模拟结果显示,所提出的不相交多路径 QoS 路由算法总是收敛的,且当网络流量增加时,该算法具有较低的包丢失率和较高的吞吐量。

关键词 不相交路径,多路径路由,带宽,时延

A Disjoint Multipath QoS Routing Algorithm in an End to End Network

ZHU Shang-Ming[1] ZHUANG Xin-Hua[2] GAO Da-Qi[1]

(Department of Computer Science, East China University of Science and Technology, Shanghai 200237)[1]

(Department of Computer Science, University of Missouri-Columbia, Columbia, MO65211, USA)[2]

Abstract Disjoint multipath routing algorithm is aimed at selecting multiple paths for a flow in an end-to-end network, which are mutually disjoint w. r. t. bottleneck links. In this paper, the proposed Disjoint Multipath QoS Routing (DMQR) algorithm is based on both Dijkstra Shortest Path and Shortest Widest Path (SWP) algorithm, and can dynamically find paths mutually disjoint w. r. t. bottleneck links. Each path is the shortest among the widest ones and its QoS is guaranteed. In many important multimedia applications such as video over IP in teleconference, telemedicine and teleeducation, the application- and network-layer must collaborate in order to provide some necessary QoS guarantee, such as end-to-end bandwidth, delay and packet loss rate, etc. In this paper, we mainly focus on a scenario of one source and one destination and design the Disjoint Multipath QoS Routing algorithm to achieve certain QoS level at the application-layer. Simulation results and performance analysis demonstrate that the algorithm converges, offers lower end-to-end packet loss rate and higher throughput as network traffic grows.

Keywords Disjoint path, Multipath routing, Bandwidth, Delay

1 引言

多路径路由是提高链路带宽利用率和端到端网络路径可靠性的一项关键技术,在基于宽带 IP 网的多媒体通信,例如视频会议、远程医疗以及远程教育等领域有着广泛的应用前景。通过路由一个应用流到多路径,可以均衡网络负载或为层叠网络(Overlay networks)提供连接冗余[1,2]。目前已提出了不少多路径路由算法,例如在流水平进行路由选择的最宽不相交路径(Widest Disjoint Paths, WDP)[3,4]和多路径负载分配(Load Distribution over Multipath, LDM)[5]算法,在包水平进行路由转发的等代价多路径(Equal Cost MultiPath, ECMP)[6]、MPLS 最优多路径(Optimized MultiPath, MPLS-OMP)[7]以及 MPLS 自适应流量工程(MPLS Adaptive Traffic Engineering, MATE)[8]算法等。

在已提出的算法中,最宽不相交路径(WDP)由于采用在瓶颈链路上互不相交的路径选择机制,在端到端的网络传输中能够提供较好的性能。但是,WDP 算法要求源-目的节点之间的所有可行路径已经静态建立,即源-目的节点之间的所有可行路径是预先已知的,这使其应用受到了限制。本文提出了一种不相交多路径 QoS 路由(DMQR)算法,该算法仍然在流水平进行路由决策,但能够动态地在源-目的节点之间建立瓶颈链路上互不相交的多个路径,并引入 QoS 的约束机制,保证选择的路径是满足一定带宽和时延约束的,且每个新加入的路径是当前可行路径集中时延最短、带宽最宽的一个。

基于宽带 IP 网的多媒体通信要求应用层和网络层必须协同工作以保证一些必要的 QoS。应用层往往需要一些经济可行的关于网络层的 QoS 性能度量,例如端到端的带宽、时延和包丢失率等,并适时地补偿网络层 QoS 性能的不足。本文主要针对端到端的网络,重点讨论不相交多路径 QoS 路由(DMQR)算法在应用层的设计实现并对其性能进行分析。

2 DMQR 算法的设计

Dijkstra 最短路径算法[9]和最短最宽路径(SWP)[10]算法均是计算从源节点到目的节点的单一最佳路径的算法。不相

*)基金项目:国家自然科学基金(No. 60373073),美国 NIH 基金(DHHS 1 R01 DC04340-01A2)和美国 NSF 基金(EIA9911095)。朱尚明 副教授,博士研究生,研究方向为计算机网络和多媒体通信;庄新华 教授,博士生导师,研究方向为多媒体通信和数字图像压缩;高大启 教授,博士生导师,研究方向为计算机网络和智能理论。

交多路径 QoS 路由(DMQR)算法则试图在 Dijkstra 最短路径算法和最短最宽路径(SWP)的基础上,在源节点与目的节点之间计算得到一个多路径集,以满足多媒体通信的 QoS 需求。该算法使用链路状态路由协议来动态地搜集和更新网络的拓扑结构和链路信息。不失一般性,本文使用带宽和时延作为一个链路或路径的 QoS 状态信息,带宽和时延也是交互式实时多媒体通信的重要指标。

不相交多路径 QoS 路由(DMQR)算法的设计思想是:多路径集中的每个路径必须保证具有一定的可用带宽和可容许时延,并且是在瓶颈链路上互不相交的。不共享瓶颈链路能够减少候选路径的数量且不会引起包丢失率的增大。但是,如何识别一个瓶颈链路是一个动态的过程,因为网络的拥塞取决于负载流量和路由选择的变化。为此,本文引入链路剩余带宽的概念。

对于一个给定的网络拓扑 G,假设具有 N 个节点,$l=(u,v)$表示节点 u 到节点 v 的一条链路,其最大带宽容量为 $C(u,v)$,是固定的且是已知的。定义 $b(u,v)$为链路 l 的剩余带宽,$d(u,v)$为链路 l 的时延。若当前已有 m 个路径经由链路 l,那么其剩余带宽为 $b(u,v)=C(u,v)-\sum_{i=1}^{m}B_i$,其中 B_i 为经由链路 l 的第 i 条路径的带宽。

设 S 为端到端通信的源节点,F 为目的节点,h 是从源节点 S 开始的跳数。再设 $b(u,h,p)$和 $d(u,h,p)$分别为从源节点 S 到当前工作节点 u 在 h 跳时一个路径 p 的带宽和时延,设 $N_0(u)=\{v_1, v_2, \cdots, v_K\}$为工作节点 u 的邻近节点,且它们之间的带宽为 $b(u,v_k)$,时延为 $d(u,v_k)$,其中 $k=1, 2, \cdots, K$。如果路径 p 可以经由 u 延伸至 v_k,可以通过下式分别计算下一跳的带宽和时延:

$$b(v_k,h+1,p)=min\{b(u,h,p),b(u,v_k)\} \quad (1)$$

$$d(v_k,h+1,p)=d(u,h,p)+d(u,v_k) \quad (2)$$

显然,路径的带宽即是其瓶颈链路上的带宽,路径的时延即是各个链路上的累积时延。该算法初始设置为:$h=0$; $d(S,.,.)=0$; $b(S,.,.)=\infty$; $d(u,0,.)=\infty$; $b(u,0,.)=0$ where $u\neq S$。

假设 $U_0(h)$是在 h 跳时从源节点 S 可达的工作节点的集合,对于 $U_0(h)$中的每个工作节点 u,如果它不是目的节点 F,该算法将检查它相应的链路状态信息,即链路带宽和链路时延,以便决定能否进行扩展。假设在 h 跳时到达工作节点 u 的路径有多个:$p_j(j=1, 2, \cdots, J)$,且每个路径的带宽和时延分别为 $b(u,h,p_j)$ 和 $d(u,h,p_j)$。对于 $k=1, 2, \cdots, K$ 和 $j=1, 2, \cdots, J$,可以根据上面的(1)和(2)式来计算$(h+1)$跳的带宽 $b(v_k,h+1,p_j)$和时延 $d(v_k,h+1,p_j)$。

由于节点 v_k在 $(h+1)$ 跳时可能会经由多个节点可达,该算法将检查在$(h+1)$跳时从源节点 S 到达节点 v_k 的所有路径,先从中删除循环的路径,然后比较所有剩余路径的带宽,挑选出一个具有带宽最宽的路径子集,再从其中选出时延最短的路径,并加以记录和标识。当节点集合 $U_0(h)$中的所有节点都被计算和扩展后,跳数 h 的值将加 1,并重复上述步骤直到 h 的值达到了网络拓扑的直径或一个给定的阈值。

为了满足 QoS 约束并实现瓶颈链路上的互不相交,对于所有从源节点 S 到达目的节点 F 的的路径集 P,将丢弃那些带宽小于可用带宽 B_0 或时延大于可容许时延 D_0 的路径。从余下的路径中选取出一个时延最短、最宽带宽的路径 p^*,并将其加入到多路径结果集 P^*。然后去除路径集 P 中包含所选路径 p^* 的瓶颈链路的路径,以实现瓶颈链路上的互不相交。以上过程重复执行,直到路径集 P 为空或路径集 P^* 的个数达到一定的阈值。这样路径集 P^* 中的路径就是从源节点 S 到目的节点 F 的在瓶颈链路上互不相交、且满足带宽和时延服务质量约束的最短最宽路径。

更进一步,还可以计算路径集 P^* 中每个路径端到端的包丢失率,以便应用层对路径进行评估和选择。也可以通过采用多重连接(Multihoming)机制来传输应用流,以保证所选择的路径在物理上是独立的或互不相交的。

3 DMQR 算法的实现

上面所介绍的不相交多路径 QoS 路由(DMQR)算法能够为一个源-目的节点之间的应用流选取满足 QoS 约束且在瓶颈链路上互不相交的多个路径。为了提高计算效率,DMQR 算法基于 Dijkstra 算法和递归计算的思想对路径进行逐跳扩展。DMQR 算法的实现过程如图 1 所示。

```
1: B0=Min_Bandwidth; D0=Max_Delay;
2: P0=Max_paths; M0=Num_multipaths; H0=Max_hops;
3: for (i=0;i< M0;i++)
4:    for each node u in G
5:       for (h=0;h< H0;h ++)
6:          for (j=0;j< P0;j++)
7:             if u=S
8:                bandwidth[S,h,j]=∞;delay[S,h,j]=0;
9:                path[S,h,j]=S; label[S,h]="t";
                  ingress(S,h)=1;
10:            else
11:               bandwidth[u,h,j]= 0; delay[u,h,j]= ∞;
12:               path[u,h,j]="";label[u,h]="" ;
                  ingress(S,h)=0;
13:   for (h=0;h< H0;h++)
14:    for each u in U0(h)
15:       If label[u,h]="t" and u≠F
16:          for each path j ingressing at node u
17:             V={ v:v∈ N0(u) and v∉ path(u,h,j) };
18:             for each node v in V
19:                g=ingress(v, h+1)+1;
20:                b[v,h+1,g]=min(b[u,h,j],b[u,v]);
21:                d[v,h+1,g]=d[u,h,j]+d[u,v];
22:                If b[v,h+1,g]≥B0 and d[u,h+1,g] ≤D0
23:                   bandwidth[v,h+1,g]= b[v,h+1,g];
24:                   delay[v,h+1,g]= d[v,h+1,g];
25:                   path[v,h+1,g]= path[u,h,j] ∪ v;
26:                   label[v,h+1]="t";
27:                   ingressing(v, h+1)=g;
28:          label[u,h]="p";
29:   P={ p : p∈ path[F,,] };
30:   P*={ p:p∈P, bandwidth[F,,] =
                  max p∈P bandwidth[F,,p] };
31:   p*={ p:p∈P*,delay[F,,]=
                  min p∈P*delay[F,,p] };
32:   for each (u,v) in p*
33:      b[u,v]= b[u,v] – bandwidth[F,, p*] ;
34:   Multipath[i]= p*;
```

图 1 算法实现

为了保证 QoS 约束，初始化 B_0 为所需要的最小带宽，D_0 为可容许的最大时延，P_0 为两个节点之间的最多路径个数，M_0 为所选择的多路径个数，H_0 为所允许的最大跳数（line 1—2）。对每一个多路径，该算法将分别进行一次迭代计算（line 3）。为了将一个路径从源节点扩展到目的节点，每个节点在每一跳时标记有若干状态信息：从源节点流入的每一个路径的瓶颈带宽 *bandwidth* 和累积时延 *delay*，构成每个路径的节点集合 *path*，每个节点的分类标识 *label* 和每个节点的流入路径个数 *ingress*。节点分类标识分为永久节点（"*p*"）和试探节点（"*t*"）两种。对每一跳，当一个节点在路径扩展时被探询到后，将被标记为试探节点；当一个节点的所有从源节点流入路径被扩展后，将被标记为永久节点且之后不再改变。每个节点在每一跳时每个路径的初始设置为带宽=0，时延=∞。而源节点要设置为根节点并标识为试探节点（"*t*"），其在每一跳时每个路径的初始带宽=∞，时延=0（line 4—12）。

接下来计算在每一跳时每个节点的状态信息（line 13—28）。对于每一跳，如果一个新扩展的节点 u 不是目的节点 F（line 13—15），该算法将检查其每个流入路径及其邻近节点 v（line 16—18），并将满足 QoS 约束条件的路径从 u 扩展到其邻近节点 v。假设 g 为邻近节点 v 的流入路径个数（line 19），对于一个应用流，从源节点 S 到当前节点 u 的一个路径 p_j 如果被扩展到邻近节点 v，在节点 v 处，其带宽为 $b[v,h+1,g]=\min\{b[u,h,j],b[u,v]\}$，时延为 $d[v,h+1,g]=d[u,h,j]+d[u,v]$（line 20—21）。

新计算的从源节点 S 到节点 v 的带宽和时延将与所要求的带宽阈值 B_0 和所容许的时延阈值 D_0 进行比较，以保证 QoS 约束（line 22）。如果不满足 QoS 约束，新计算的路径将被丢弃，否则将更新节点 v 的状态信息，记录下新的路径带宽、时延和节点集合，将 v 的标识为试探节点（"*t*"），并修改其流入路径数量（Line 23—27）。当节点 u 的所有邻近节点都被探询之后，在当前 h 跳的计算中，节点 u 将被标识为永久节点（"*p*"），并从当前工作节点中去除（line 28）。

在目的节点 F，将进行最短最宽路径的选择，首先计算一个具有带宽最宽的路径子集，再从其中选出时延最短的路径（line 29—31）。为了实现路径在瓶颈链路上的互不相交，每次选出一个路径之后将对构成该路径的每条链路的带宽进行修正，计算出其剩余带宽（line 32—33）。最后将所选择的路径输出到数组 Multipath[i]中（line 34）。迭代计算的次数取决于需要选择的多路径的个数。

4 性能分析

4.1 复杂度分析

多路径 QoS 路由算法的成功取决于如何进行多路径的选择，所选择的路径数量和服务质量将决定路由机制的性能和效率。众所周知，一个路径的建立、维持和拆除将会给网络造成一定的开销，随着网络节点和链路数量的增大，多路径计算的复杂度也会随之增加。

由于本文提出的不相交多路径路由（DMQR）算法能够避免路径循环，且基于 Dijkstra 算法的思想进行路径扩展，计算次数受限于最大跳数 H_0，对于一个给定的网络，该算法对所有节点总是收敛的。DMQR 算法的计算复杂度相对较低，计算效率也较高。由于 Dijkstra 算法计算一个路径的复杂度是 $O(E+N\log N)$，其中 E 是网络中链路的个数，N 是网络中节点的个数，那么 DMQR 算法计算 m 个多路径的复杂度为 $O(m(E+N\log N))$。此外，如果不考虑 QoS 约束（$B_0=0$，$D_0=\infty$），并且仅考虑选择一个路径（$M_0=1$），上面所描述的 DMQR 算法将退化为最短最宽路径（SWP）算法。

4.2 模拟结果

为了分析所提出的不相交多路径 QoS 路由（DMQR）算法的性能，我们对算法进行了模拟实验。模拟实验的网络模型如图 2 所示。每条链路的带宽和时延值已标注在链路边上，单位分别为 Mbps 和 ms。网络中的节点编号为从 ① 到 ⑨。

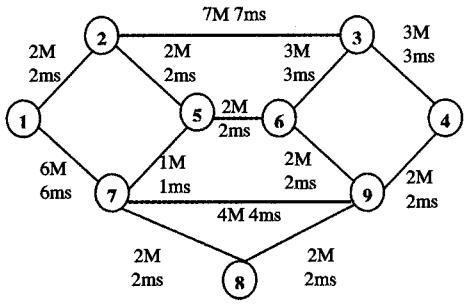

图 2 网络模拟模型

本文采用 C/C++ 语言对不相交多路径 QoS 路由（DMQR）算法、Dijkstra 最短路径和最短最宽路径（SWP）算法进行了模拟实现。模拟实现时，定义节点①为源节点，节点⑨为目的节点。最小可用带宽设置为 $B_0=2$Mbps，最大容许时延设置为 $D_0=10$ms。

DMQR 算法在 $M_0=1,2,3$ 时的选择结果以及与 Dijkstra 最短路径和最短最宽路径（SWP）算法的选择结果对比如表 1 所示。从中可以看出，当 $M_0=1$ 时 DMQR 算法的选择结果与最短最宽路径（SWP）算法的选择结果是相同的，即时延最短、带宽最宽的一条路径（1 7 9），而 Dijkstra 最短路径算法输出的仅是时延最短的一条路径（1 2 5 6 9）。

为了进行性能分析，本文采用 NS-2 进行仿真模拟，并使用端到端的包丢失率和吞吐量作为算法的度量尺度，比较其随包发送速率的变化。简单起见，设置所选择的多路径个数 $M_0=2$，发送的 UDP 包尺寸为 1000，仿真实验结果如图 3 和图 4 所示。

图 3 给出了包丢失率随包发送速率变化的模拟结果。从中可以看出，与 Dijkstra 最短路径和最短最宽路径（SWP）算法相比，DMQR 算法由于采用多路径 QoS 路由，具有较低的包丢失率；而且随着包发送速率的增大，其包丢失率的增加速度也是最慢的。同时还可以看出，Dijkstra 和最短最宽路径（SWP）算法由于都是采用单一最佳路径来路由包，其包丢失率的变化斜率是近似的，不过对于相同的包发送速率，最短最宽路径（SWP）的包丢失率要低于 Dijkstra 最短路径算法。

图 4 给出了网络吞吐量随包发送速率变化的模拟结果。从中可以看出，仅当包发送速率较低时（<2M），Dijkstra 最短路径算法性能稍优于最短最宽路径（SWP）算法和 DMQR 算法；随着包发送速率的增大，DMQR 和最短最宽路径（SWP）算法由于采用了较宽的路径来路由包，因此具有较高的吞吐量；而且随着网络流量的进一步增加，DMQR 算法由于采用了互不相交的多路径进行路由，在网络吞吐量方面的优势更为明显。

表 1 路径选择结果

Algorithm	path output	Bandwidth	Delay	Hops
DMQR (1)	path1: 1 7 9	4M	10ms	2
DMQR (2)	path1: 1 7 9	4 M	10ms	2
	path2: 1 2 5 6 9	2 M	8ms	4
DMQR (3)	path1: 1 7 9	4 M	10ms	2
	path2: 1 2 5 6 9	2 M	8ms	4
	path3: 1 7 8 9	2 M	10ms	3
Dijkstra	path: 1 2 5 6 9	2 M	8ms	4
SWP	path: 1 7 9	4 M	10ms	2

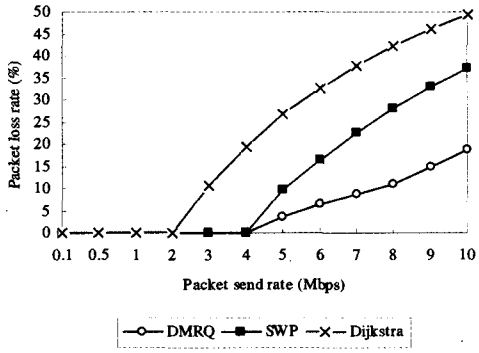

图 3 包丢失率比较

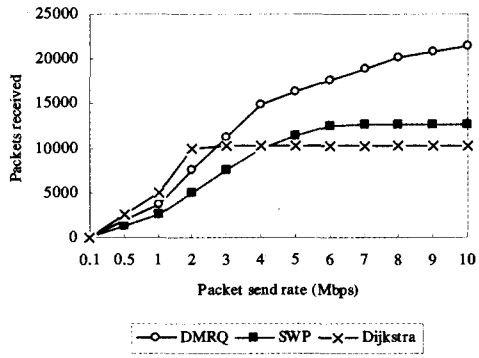

图 4 网络吞吐量比较

结束语 本文针对端到端的网络，提出了一种不相交多路径 QoS 路由(DMQR)算法，该算法在满足带宽和时延的 QoS 约束下，基于 Dijkstra 最短路径算法对路径进行扩展，基于最短最宽路径(SWP) 算法进行选路，并保证选择的路径在瓶颈链路上是互不相交的。性能分析表明，所提出的不相交多路径 QoS 路由(DMQR)算法总是收敛的，并具有较低的计算复杂度。模拟结果显示，随着网络负载流量的增大，该算法与 Dijkstra 最短路径和最短最宽路径(SWP) 算法相比，具有较低的包丢失率和较高的网络吞吐量。

本文的研究目的是试图建立一个 QoS 路由的多路径选择机制，其研究结果尚需通过实际的网络环境来验证。通过实际的网络环境，可以对采用该机制的端到端通信中的时延、抖动、包丢失率等进行更深入的研究和分析。

参 考 文 献

1 Lee K, Toguyeni A, Noce A, et al. Comparison of multipath algorithms for load balancing in a MPLS network. ICOIN 2005, Springer-Verlag Berlin Heidelberg, LNCS 3391, 2005. 463～470

2 Akella A, Pang J, Shaikh A, et al. A comparison of overlay routing and multihoming route control. In: Proceedings of ACM SIGCOMM '04, Portland, Oregon, Aug. 2004. 93～106

3 Nelakuditi S, Zhang Zhi-Li, Du D H C. On selection of candidate paths for proportional routing. Computer Networks, 2004, 44: 79～102

4 Nelakuditi S, Zhang Zhi-Li. On selection of paths for multipath routing. In: IWQoS 2001 : 9th International Workshop Karlsruhe, Germany, Proceedings, Volume 2092, 2001. 170～184

5 Song Jeonghwa, Kim S, Lee M. Dynamic Load distribution in MPLS networks. ICOIN 2003, Springer-Verlag Berlin Heidelberg, LNCS 2662, 2003. 989～999

6 Moy J. OSPF version 2, RFC 2328. Internet Engineering Task Force, April 1998

7 Villamizar C. MPLS optimized multipath (MPLS-OMP). Work in progress Internet-Draft ⟨draft-villamizar-mpls-omp-01⟩, Feb. 1999

8 Elwalid A, Jin C, Low S, et al. MATE: MPLS adaptive traffic engineering. INFOCOM'2001, Alaska, 2001

9 Tanenbaum A S. Computer Networks (fourth edition). Prentice Hall PTR, Aug. 2002

10 Wang Z, Crowcroft J. QoS routing for supporting multimedia applications. IEEE Journal of Selected Areas in Communications, 1996, 14: 1228～1234

(上接第 26 页)

问吞吐量；同时，仿真结果也表明连接时间估计准确性对算法的重要影响，在以后的工作中我们将继续深入研究这一问题，以进一步提高算法的性能。

参 考 文 献

1 Akyildiz I F, Wang X, Wang W. Wireless Mesh Networks: A Survey. Computer Networks Journal (Elsevier), March 2005, 47: 445～487

2 Mishra A, Shin M. Arbaugh W. An empirical analysis of the IEEE 802.11 MAC layer handoff process: [Technical Report]. UMIACS-TR-2002-75. University of Maryland 2002

3 Ramachandran K, Buddhikot M M, Chandranmenon G, et al. On the Design and Implementation of Infrastructure Mesh Networks. In: Proc. WiMesh 2005, Santa Clara, CA, October 2005

4 Nguyen H, Morikawa H, Aoyama T. Personal Mesh: Realizing Flexible Internet Access for Personal Area Network. In: Proc. 11th Asia-Pacific Conference on Communications (APCC), October 2005

5 Amir Y, Danilov C, Hilsdale M, et al. Fast Handoff for Seamless Wireless Mesh Networks. In: the International Conference on Mobile Systems, Applications and Services (MobiSys 2006), Upsala, Sweden, June 2006

6 Draves R, Padhye J, Zill B. Routing in Multi-radio, Multi-hop Wireless Mesh Networks. In: ACM MobiCom, Philadelphia, PA, September 2004

7 The network simulator - ns-2. http://www.isi.edu/nsnam/ns/, 2002

8 Perkins C E, Bhagwat P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proc. SIGCOMM'94. New York: ACM Press, 1994. 234～244

9 Velayos H, Karlsson G. Techniques to Reduce IEEE 802.11b MAC Layer Handover Time. In: Proc. IEEE ICC, June 2004

10 Capkun S, Hamdi M, Hubaux J. GPS-free positioning in mobile ad-hoc networks. In: Proc. 34th IEEE HICSS, 2001. 3481～3490

11 Su W, Lee Sung-Ju, Gerla M. Mobility prediction and routing in ad hoc wireless networks. International Journal of Network Management, 2001

12 Jardosh A, Belding-Royer E M, Almeroth K C, et al. Towards realistic mobility models for mobile ad hoc networks. In: Proc. 9th annual international conference on Mobile computing and networking, September 2003

13 Camp T, Boleng J, Davies V. A Survey of Mobility Models for Ad Hoc Network Research. Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking Research, Trends and Applications, 2002, 2(5): 483～502