Designing Wireless Radio Access Networks for Third Generation Cellular Networks

Tian Bu
Bell Laboratories
Lucent Technologies
Holmdel, New Jersey 07733
Email: tbu@dnrc.bell-labs.com

Mun Choon Chan
Dept. of Computer Science,
NUS
Singapore
Email: chanmc@comp.nus.edu.sg

Ramachandran Ramjee
Bell Laboratories
Lucent Technologies
Murray Hill, New Jersey 07974
Email: ramjee@research.bell-labs.com

***Abstract*— In third generation (3G) cellular networks, base stations are connected to base station controllers by point-to-point (usually T1/E1) links. However, today's T1/E1 based backhaul network is not a good match for next generation wireless networks because symmetric T1s is not an efficient way to carry bursty and asymmetric data traffic. In this paper, we propose designing an IEEE 802.16-based wireless radio access network to carry the traffic from the base station to the radio network controller. 802.16 has several characteristics that make it a better match for 3G radio access networks including its support for Time Division Duplex mode that supports asymmetry efficiently.**

In this paper, we tackle the following question: given a layout of base stations and base station controllers, how do we design the topology of the 802.16 radio access network connecting the base stations to the base station controller that minimizes the number of 802.16 links used while meeting the expected demands of traffic from/to the base stations? We make three contributions: we first show that finding the optimal solution to the problem is NP-hard. We then provide heuristics that perform close to the optimal solution. Finally, we address the reliability issue of failure of 802.16 links or nodes by designing algorithms to create topologies that can handle single failures effectively.

I. Introduction

Third Generation (3G) wireless networks based on Code Division Multiple Access (CDMA) technology are now being increasingly deployed throughout the world. As of March 2004, there were over 200 million CDMA subscribers worldwide [1]. 3G CDMA networks such as CDMA2000 1X allows doubling of voice capacity over regular CDMA and CDMA2000 EV-DO supports high-speed wireless data with peak rates of up to 2.4 Mbps.

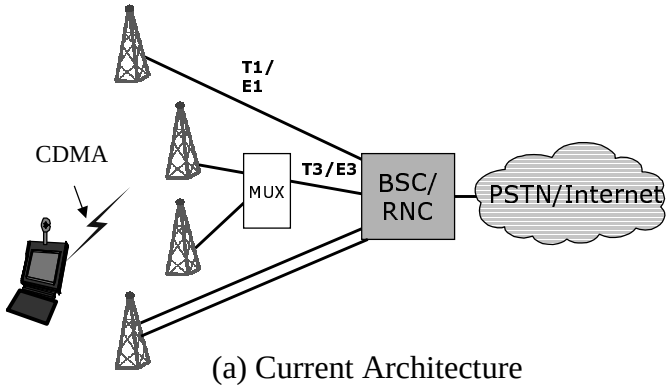
In these third generation cellular networks, the base stations are connected to radio network controllers or base station controllers by point-to-point (usually T1/E1) links as shown in Figure 1(a). These links, also called backhaul links, are expensive and their use imposes an on-going cost on the service providers. As more of the current CDMA subscribers migrate to higher capacity CDMA2000 and high-speed wireless data based on CDMA2000 EV-DO, the current radio access network will increasingly become a bottleneck, forcing service providers to add more of these costly T1/E1 links to support the higher capacity air interface.

In addition, today's T1/E1 based backhaul network is not a good match for third generation wireless networks due to the following reasons: 1) T1/E1, which provides symmetric bandwidth in both uplink and downlink, while a good fit for carrying voice traffic, is not well suited for bursty and asymmetric data traffic; 2) T1/E1s are a source of reliability problems as adding redundancy through additional point-to-point links is expensive; 3) T1/E1 provisioning can take significant time (in some cases, months) limiting the service providers ability to react quickly to changing demands.

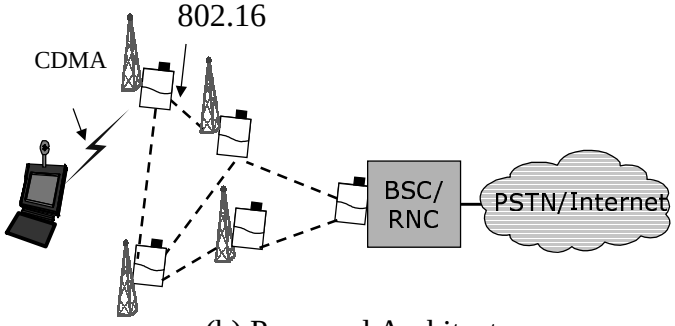
In this paper, we propose designing a wireless radio access network to carry the traffic from the base stations to the radio network controller. While fixed wireless systems based on LMDS/MMDS technology have been around for decades, they haven't been really successful in backhaul applications to date since they have been based on proprietary technologies that results in lock-in and high cost. However, with the recent ratification of the IEEE 802.16 wireless metropolitan area network (WirelessMAN) standard [2], the cost of fixed wireless systems should be dramatically lower. According to WiMAX forum, an 802.16 Base Station should cost less than $20000 and an 802.16 Subscriber Station should cost less than $300 [3], eventually reaching the same cost as an 802.11 card ($30) .

Apart from the low cost, several of the 802.16 features appear a good fit for the wireless radio access network application. IEEE 802.16 has support for Time Division Duplex (TDD) mode that enables dynamically adjusting to bursty and asymmetric data traffic. IEEE 802.16 has both point-to-multipoint and multi-hop mesh support. Mesh support is extremely useful when the base station controller is out of the range of a base station but can be reached in a multi-hop manner through other intermediate base stations. IEEE 802.16 base station controls and allocates resources to subscriber stations in both uplink and downlink, thus enabling the support for multiple quality of service classes (a necessity in tightly synchronized CDMA networks). IEEE 802.16 operates in both line-of-sight and non-line-of-sight modes, thus allowing deployments in regions where there is no direct line-of-sight. Finally, managing uncertain demands is easy as the service provider can enable more on-site 802.16 links dynamically rather than wait for T1/E1 provisioning.

In this paper, we tackle the following question: given a network layout (base stations and base station controllers), what is the optimal algorithm for designing the topology of the 802.16 radio access network connecting the base stations to the base station controller that minimizes the number of 802.16 links used while meeting the expected demands of traffic from/to the base stations? We make three contributions:



(a) Current Architecture



(b) Proposed Architecture

Fig. 1. Architecture

we first show that finding the optimal solution to the problem is NP-hard. We then provide algorithms that come close to the optimal solution. Finally, we address the reliability issue of failure of 802.16 links or nodes by designing algorithms to create topologies that can handle single link or node failures effectively.

The rest of the paper is structured as follows. In Section II, we present background and related work. Section III, we motivate the benefits of an 802.16 radio access network. In Section IV, we formally define the topology design problem and show that finding the optimal solution to the problem is NP-hard. In Section V, we provide an integer program formulation to the problem and detail our greedy algorithm for designing the 802.16-based radio access network. In Section VI, we address the issue of topology design that are resilient to single link and node failures by showing that the optimal solutions to these problems are NP-hard and detail our greedy algorithm for solving this problem. In Section VII, we evaluate our algorithms through simulations based on the base station layout of a large service provider in the United States and show that the algorithm delivers performance close to the optimal solution. Finally, in Section VIII, we present our conclusions.

II. Related work

One effective way to reduce radio access network costs is to replace the point-to-point links in the current architecture with an IP-based Radio Access Network [4], [5], [6] (IP-based RAN). This allows statistical multiplexing of traffic within the RAN resulting in cost savings as long as appropriate QoS can be ensured. Several researchers have proposed solutions for addressing quality of service (QoS) issues in IP-based RANs [7], [8], [5]. The question of connectivity, i.e. how best to connect base stations to the radio network controllers in the IP-based RAN, was addressed by [9]. However, all these solutions assume that the IP-based RAN consists only of wired links. Wireless connections based on 802.16 allow the operator to connect the base stations together directly, resulting in even lower costs than a wired IP-based RAN.

Problems similar to the ones tackled in this paper have been considered in other contexts. For example, researchers have looked at problems of reducing the wiring cost in a wireline network. [10] assumes there is only one cable type of bandwidth k and there is at most one cable between a pair of nodes. The cost of a cable is proportional to the distance between two nodes that it connects. There is a root node and a set of demands from other nodes to the root node. The goal is to find a minimum steiner tree rooted at r such that the total demands of any subtree is not greater than k. [11] looks at a generalized version of the problem where there are multiple cable types of different costs and each pair of nodes can be connected by multiple cables of different types. The goal is to minimize the total cable costs without limiting the solution to be a tree. Although the objectives of the two problems are close to ours, that is to reduce the cost of connecting nodes meeting a given set of demands, our problem of minimizing wireless links is quite different from minimizing wiring cost in wireline network. In the wireless case, we do not have the choices of different cable types between a pair of nodes. In fact, the capacity of a wireless link between a pair of nodes is a function of distance and transmission power. In addition, the cost of a wireless link does not depend on the distance but is equal to the fixed cost of installing a transmitter/receiver at both ends of the link.

The link/node failure resiliency problem considered in this paper is closely related to the problems of finding the 2-edge connected subgraph and the 2-vertex connected subgraph that are all NP-hard problems. However, the problems considered in this paper are even harder to approximate due to the capacity constraints of wireless links. A 2-edge (2-vertex) connected graph is not necessarily resilient to single link (node) failure in our case. There is work [12] to find the 2-edge (2-vertex) connected graph accounting for capacity constraints. It takes the same network and demand models as in [10] and asks for a minimum cost spanning network such that the removal of the root node and its incident edges breaks the network into a number of components, each of which is 2-edge (2-vetex) connected with a total demand k. However, these wireline resiliency problems again have different constraints. It assumes that all the cables are of the same type and of same capacity k and the cost of connecting two nodes is proportional to the distance.

III. Motivation

There are two main performance benefits to replacing the current point-to-point T1/E1-based backhaul with a 802.16-based wireless radio access network. First, 802.16 has support for Time Division Duplex which allows adapting to asymmetry in uplink/downlink traffic while the symmetric nature of T1/E1s require capacity provisioning that can handle the maximum of uplink/downlink traffic. Second, a mesh-based

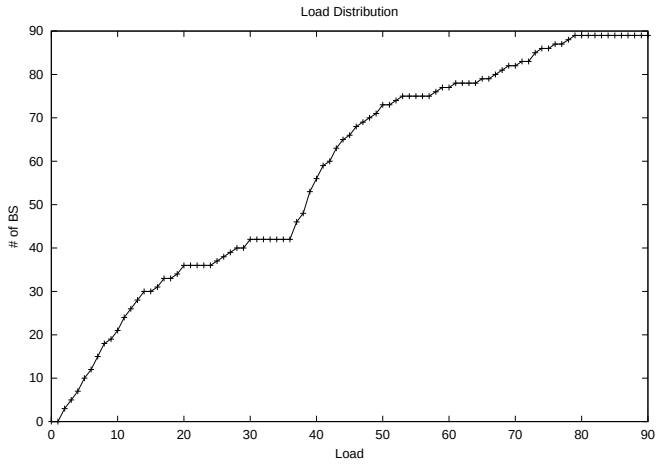


Fig. 2. Peak demand across different base stations

802.16 radio access network allows for sharing of links that is not feasible in current point-to-point networks.

As shown in Figure 1(a), the current radio access network consists of point-to-point T1/E1 links - this results in poor utilization of resources as these links cannot be shared between base stations. Direct wired connectivity between base stations is typically not feasible given constraints as to where cables are laid; logical wired connectivity between two base stations is possible but this would typically take a physical paths that would go from one base station to a central office back to the second base station, resulting in much longer physical path and higher costs; thus, architecture is Figure 1(a) is typically adopted today.

In the case of a wireless radio access network, two base stations can easily be directly connected by a wireless link. Furthermore, given that cellular base stations are mounted on high towers (typically 30 meters above the ground) for better cellular coverage, line-of-sight is typically not an issue and directional antenna can be employed to increase the transmission range and/or throughput. Direct connectivity is thus not possible only when the wireless signal is not reachable from one base station to another because of either significant distance between the two nodes or the presence of a large obstacle that dissipates the wireless signal. These cases can be handled by a multihop wireless network allowing selected base stations to connect to the RNC by using other base stations as relay nodes. Additional relay nodes can be added such that all base station can be connected to RNC as shown in Figure 1(b).

In order to quantify the benefits of the statistical multiplexing in the proposed architecture over the current architecture, we conducted the following experiment. We obtained data from a study [13] that had the load distribution for 90 base stations. While that study was for users of cellular voice service, we extrapolated the voice usage to obtain a similar distribution for wireless data load. Each user's data requests were modeled assuming a simple web session with exponentially distributed think times. The user power-up/power-down behavior was dictated by the model from [14] based on the Metricom network. We further assumed that each data event corresponded to the use of a single unit of resource. The cumulative peak load distribution for the 90 base stations is shown in Figure 2. There are 56 base stations with peak load less than or equal to 40 resource units and all base stations have peak load 80 units or less. It is clear from the figure that there is significant variation in load among the different base stations.

If this load has to be accommodated by using the current architecture where there is no sharing among base stations, we would require radio access network capacity of 2885 units. This is the sum of all peak demands over all base stations. However, if all the demands can be satisfied by a single-hop 802.16 based architecture, then only 1525 units of capacity is necessary. This is the peak demand of all base stations at any time and is much smaller since the load from different base stations can be time-multiplexed on this "common" 802.16-based backhaul. However, the pure approach assumes that the RNC is reachable from every base station which may not be feasible in some cases due to transmission range limitations. We can use the multi-hop 802.16 mesh network for these case where some base stations use other base stations as relay to reach the RNC. This might increase the channels needed. However, since the 3G base stations are usually placed not very far from each other in order to provide good coverage to an area, (see Figure 9 for the layout of a real deployment), the additional channels required for multi-hop routing is minimal.

Thus, this section motivates our proposed architecture by illustrating the benefits of having an IEEE 802.16 based architecture for the radio access network as it supports asymmetry and enables sharing of resources. This can result in significant savings in backhaul costs, especially with bursty and asymmetric data traffic. However, in order to obtain this benefit, we still need to solve the problem of how best to connect the base stations to the RNC using 802.16 links such that the traffic demands are satisfied. This is explored in more detail in the rest of paper.

IV. Models and Assumptions

In this section, we describe our model for an IEEE 802.16 network that is used as backhaul for 3G wireless network.

We consider a multi-hop 802.16 network where the nodes communicate with each other via wireless links. Each node in a network can communicate directly with a subset of the other nodes in the network. We use a bidirectional link $e =< u, v >$ to represent the fact that node v and node u is within the transmission range of each other. Since 802.16 allows dynamicl allocation of bandwidth between two directions of one wireless link, we are only interested in the total. Therefore, we denote $r(e)$, the transmission rate of wireless link e, which is the sum of transmission rates on both directions. Let us denote the network by a graph $G = (V, E)$ where V represents the set of nodes in the network and E the set of wireless links in the network. Following the 802.16 protocol, we assume that system operates in a synchronous time-slotted mode, where the length of a time-slot is τ seconds.

We assume that 802.16 nodes are stationary (mounted on 3G base station towers) and there is no necessity to employ dynamic power control. Thus, given a pair of fixed nodes and stationary channel conditions, the capacity of the link between the nodes is pre-determined and the link is always activated at this rate using the optimal power level.

In order to achieve the high rate required by the 3G backhaul, we further assume that the directional antenna is used where possible as described in the previous section. Thus, different 802.16 wireless links do not interfere with each other; side-lobe interference can be avoided by placing the different links at different heights in the 3G tower.

In the backhaul of 3G data network, traffic typically traverses only between the RNC and different base stations (BS). We locate the 802.16 base station with the RNC and an 802.16 subscriber station is co-located with every 3G base station. We will refer the 802.16 base station as root node in the graph from now on. For a given subscriber station v, there is a demand d_v to reach subscriber station v from the 802.16 base station. We let $D = (d_1, d_2, \ldots, d_{|V|})$ denote all demands. For bursty data applications, the downlink traffic (from RNC to 3G base station) results in significantly higher bandwidth usage than uplink traffic (for e.g., CDMA EV-DO has a peak rate of 2.4 Mbps downlink but only 384Kbps uplink). Thus, for the rest of this paper, *we assume that the demands d_v represent peak demand in the downlink direction* and the uplink demands can be supported by appropriately sizing the Time Division Duplex (TDD) behavior of 802.16, as long as the total demand (uplink and downlink) on every link is bounded by $r(e)$. Further, the quality of service support in 802.16 can be used to appropriately service delay sensitive traffic - for example, the voice traffic can be scheduled using the TDM mode of 802.16, resulting in no delay jitter.

Note that the deployment cost of the proposed 802.16 network is mainly the sum of the cost of 802.16 base/subscriber stations and the cost of setting up each of the links.Since the 802.16 stations are co-located with 3G base stations, they incur very little additional cost. Each wireless link generally incurs a fixed cost for the installation of the transmitter/receiver pair. Therefore, the cost of using 802.16 as radio access network is mainly determined by how many wireless links needed to be set up. In this paper, *we consider the problem of finding an 802.16 network with minimum number of links that can achieve the given demands*. Formally speaking, we want to find $G' = (V, E')$ where $E' \subseteq E$ such that $|E'|$ is minimized subject to demands d_v for all $v \in V$ being met. We will refer to this as the Minimum Links Problem (MLP).

Theorem 1: Finding an 802.16 network of minimum number of links that can achieve a given set of demands is NP-hard.

Proof: This can be proved by reducing the bin-packing problem [15] to it. In the bin-packing problem, the goal is to determine how a given number of objects (of different sizes) can be placed into the least number of fixed space bins. There are several variants of the problems that are all NP-hard. We reduce the following version of bin packing problem to the MLP. Let define a set of objects with size $s_1, s_2, \ldots, s_n$ and a set of bins of capacity c. The number of bins is not bounded. The goal is to put all objects into the least number of bins. Obviously we have $\forall i, s_i \leq c$. For such a bin-packing problem, we construct a graph as shown in Figure 3. We create n nodes $b_1, \ldots, b_n$ and attach them to node 0. The capacity of link $< 0, b_i >, 1 \leq i \leq n$ is c. In addition, for all object i with size of s_i, we create a node with demand s_i. Node 0 is the source for all demands. We connect $s_i, \forall i$ to all b_j with links of infinite capacity. If we assume that a demand can only be carried by a single flow, the solution of finding a sub-graph of

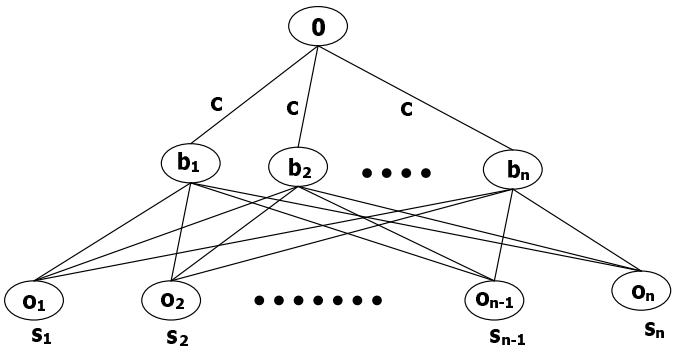


Fig. 3. Bin packing reduction

the least number of links that can deliver the given demands implies a solution of the corresponding bin-packing problem. In this case, the degrees of nodes $o_i, 1 \leq i \leq n$ have to be exactly one. Minimizing the total number of links can only be done by minimizing the number of links connecting node 0 and $b_j, 1 \leq j \leq n$. Since each such link corresponds to a bin, minimizing the number of these links is to minimize the number of bins. ■

However, the solutions of MLP allows that a demand can be carried by multiple flows taking different paths. We now further show that the best solution for the particular problem reduced from bin-packing is obtained when every demand is delivered by a single flow. Assume demand of node o_i is delivered by multiple flows in the best solution, i.e., there are multiple b_j connected to node o_i. We can convert this to the unsplitable flow version by picking a different b_k such that the residual capacity on link $< 0, b_k >$ is greater than s_i. We replace all the links connecting to s_i by links $< b_k, s_i >$ and $< 0, b_k >$ (if it is not used by other flows). We now have a solution with the same or less number of links.

V. Minimum link topologies

In this section, we first demonstrate how to formulate MLP as an 0-1 integer programming problem. Since the integer programming cannot be solved in polynomial time, we present a greedy algorithm that delivers close to optimal solution.

A. 0-1 integer programming

Let p denote a path that is consist of a set of links. P_v denotes all the possible paths from root node to node v. y_p is the amount of traffic carried by path p. x_e is a binary variable that is 1 when x_e is in the set of least number of links otherwise 0. The MLP problem can be formulated as the following integer program

$$\begin{aligned}
\min \quad & \sum_{\forall e \in E} x_e \\
\frac{\sum_v \sum_{p \in P_v : e \in p} y_p}{r(e)} \quad & \leq \quad x_e \qquad \forall e \in E \\
\sum_{p \in P_v} y_p \quad & \geq \quad d_v \qquad \forall v \\
y_p \quad & \geq \quad 0 \qquad \forall p \in P_v, \forall v \\
x_e \quad & \in \quad \{0, 1\} \qquad \forall e \in E
\end{aligned}$$

PickLeastLinks ($G' = (V, E_s), D$)

1. $\forall v \in V \setminus \{s\}$
   find P_v, set of all paths in G' from s to v
2. $u(e) = \sum_{v \in V} \sum_{e \in p : p \in P_v} 1$, $\forall e \in E_s$
3. $OLD = \max_{e \in E_s} u(e)$
4. $SUM\_POPULARITY = \max_{e \in E_s} u(e) + OLD$
5. $\bar{u}(e) = SUM\_POPULARITY - u(e)$
6. Sort all $v \in V \setminus \{s\}$ as
   $v_1, v_2, \ldots, v_{|V|-1}$ s.t. $d_{v_i} \geq d_{v_j}$ if $i < j$
7. $L = \{\}$
8. **For** $i = 1$ **to** $|V| - 1$
9. Find $p_{v_i} = \min_{p \in P_{v_i}} \sum_{e \in p} \bar{u}(e)$
   and $\sum_{j \leq i : e \in p_{v_j}} d_{v_j} \leq r(e), \forall e \in E$
10. If no such p_{v_i} can be found, return $\{\}$
11. $\forall e \in p_{v_i} \setminus L$
    $\bar{u}(e) = \bar{u}(e) - OLD$
12. $L = L \cup p_{v_i}$
13. **EndFor**
14. Sort all $e \in L$ as
    $e_1, e_2, \ldots, e_{|L|}$ s.t. $r(e_i) \leq r(e_j)$ if $i < j$
15. **Do**
16. **For** $i = 1$ **to** $|L|$
17. remove link e_i from L
18. reroute all flows that traverse link e_i
19. If any flow cannot be rerouted, add e_i to L
20. **EndFor**
21. **Until** no link removed in the round
21. return L

Fig. 4. *PickLeastLinks*()

The first constraint is to ensure that the aggregated traffic from all flows traversing a links does not exceed the capacity of the link if the link is in use. Otherwise the aggregated traffic is zero. The second constraints states that the aggregated traffic among flows from root node to node v at least satisfies the demand of node v. The third constraint is to prevent a negative-valued flow and the last one ensures x_e is a binary variable.

Unfortunately, the integer programming problem can be solved for only small sized problems in practice because of the integer requirements and the exponential number of variables. We now describe our heuristic-based greedy algorithm for solving MLP. We will use the integer programming to evaluate the quality of solution yielded by the greedy algorithm when the problem size is small.

B. Greedy algorithm for MLP

We have three main preferences in choosing a path for MLP with the objective to use fewer links:

- shortest path for each individual demand
- share usage of links between different demands
- choose high capacity links over low capacity links

Note that some of these preferences are inter-related (choosing high capacity links allow more sharing etc.). On the other hand, some preferences are conflicting. A shorter path (which uses less hops) tends to join stations that are far apart, while high capacity links tend to use links joining stations that are close together.

MLP_heuristic ($G = (V, E), D$)

1. Sort all links $e \in E$ as
   $e_1, e_2, \ldots, e_{|E|}$ s.t. $e_i \geq e_j$ if $i < j$
2. $E_s = \{\}$, $\min = +\infty$
3. **For** $i = 1$ **to** $|E|$
4. $E_s = E_s \cup \{e_i\}$
5. If there is at least one path from s to all $v \in V$
6. R = PickLeastLinks($G' = (V, E_s), D$)
7. **If** $R \neq \{\}$ and $|R| < \min$
8. $\min = |R|, \min R = R$
9. **EndFor**
10. return $\min R$

Fig. 5. MLP Greedy Algorithm

The basic algorithm, *PickLeastLinks()*, computes the minimum set of links needed to support the demand D given the graph G'. It is a greedy algorithm that tries to combine the first two preferences, shorter path for each individual demand and more sharing of links among paths for different demands. The details of the algorithm is described in Figure 4. *MLP Greedy* algorithm described in Figure 5 is the main algorithm. It tries to cater to the third preference (choosing high capacity links over low capacity links) and uses the results of the basic greedy algorithm to minimize the number of links used in MLP.

In order to encourage sharing of links among different paths, we rank the links in a graph by a *popularity* metric. The popularity of a link e, denoted $u(e)$, is initially the number of paths that traverse the link among all possible paths that can carry demands (lines 1-2 of Figure 4). The intuition is that the higher the popularity of a link, the more likely it can be shared among different paths. The goal of the greedy algorithm is then to pick a path of fewer hops while using the more popular links. This is done by converting the popularity metric $u(.)$ to an unpopularity metric $\bar{u}(.) = SUM\_POPULARITY - u(.)$ and treating $\bar{u}(.)$ as the cost of links (line 5). A standard shortest path algorithm can then find the shortest path that maximizes the popularity (line 9).

Our simulation evaluation suggests that the choice of $SUM\_POPULARITY$ is not very important as long as it is greater than the sum of the maximum of all popularities and a constant OLD (explained next). Once a link has been used by any path for satisfying a demand, the unpopularity is reduced by OLD (line 11). OLD is a constant designed to encourage the greedy algorithm to pick paths traversing links that is already part of existing links. The larger the value of OLD, the less likely a new link is picked. According to our simulation results, setting OLD to be the maximum popularity (line 3) is sufficient and there is no gain in using any greater values.

Once a path has been picked for every demand, we further improve the results by pruning the chosen set of links in the previous step. This is done by first sorting the links in the descending order according to the link bandwidth (line 14) .

Starting from the link of the least bandwidth, we remove one link at a time (line 17) . Once a link is removed, we try to reroute all paths that traverse this link (line 18). If alternatives can be found for all the paths, we permanently remove the link - otherwise we add the link back (line 19). We repeat this process until some demand can not be satisfied upon the removal of any single link.

We observed during the simulation that the presence of too many low bandwidth links can result in reducing the performance of *PickLeastLinks()*. The reason is that these low bandwidth links provides too many choices to the greedy algorithm. In addition, since the bandwidth of the wireless links is inversely proportional to the square of the distance, the path that uses these low bandwidth links tends to be shorter by hop-count - this makes the link more likely to be chosen by the greedy algorithm. However, the low bandwidth links can be shared by fewer paths due to the capacity constraint which ends up using more links. We propose that the MLP Greedy Algorithm makes use of this observation by giving higher preference to high bandwidth links. We sort the links according to the link bandwidth in the decreasing order of bandwidth (line 1 of Figure 5). We use the smallest connected subgraph that contains the highest bandwidth links as the initial input graph. After each computation by PickLeastLinks(), the input graph is expanded by adding the next largest link (line 4). The minimum set of links returned by PickLeastLinks() over all the different input graphs (line 6-8) is the output of MLP Greedy algorithm.

VI. Node and link failure resistant topologies

In the previous section, we described our algorithms for finding the minimum number of links in an 802.16 network that can satisfy a given set of demands. Since both the links and the nodes can fail during network operation, in this section, we investigate how to build a network with the least number of links that is resilient to a single link or node failure. We first show that both the link-failure resistance and node-failure resistant topology design problems are NP-hard. We then formulate integer programming solutions for both problems to solve them exactly. Since the integer programming is not practical for large networks, we then propose greedy algorithms for these two problems.

For a network $G = (V, E)$ and given demands $D = (d_1, d_2, \ldots, d_v)$, our goal is to find the least number of links such that the demands can always be satisfied even when there is a single link or a single no root node failure. We refer the problems as Minimizing number of Links Resilient to Link failure Problem (MLRLP) and Minimizing number of Links Resilient to Node failure Problem (MLRNP).

Theorem 2: Finding an 802.16 network of minimum number of links that can achieve a given set of demands even when there is at most one link failure is NP-hard.

Proof: This can be proved by reducing the minimum 2-edge connected subgraph problem [15] to it. For a given graph $G = (V, E)$, the goal of the minimum 2-edge connected subgraph problem is to find a 2-edge connected spanning subgraph $G' = (V, E')$ such that the number of links $|E'|$ is minimized. Let us pick a node in G as the root node and assign demands $d_1, d_2, \ldots, d_{|v|-1}$ to the other nodes. We further assign a capacity to each link that double the sum of all demands. The solution of finding the minimum number of links that can satisfy the demands even when there is a single link failure implies a solution to the minimum 2-edge connected subgraph problem. Obviously if a graph is not 2-edge connected, our resilience requirement cannot be met. On the other hand, if the resilience is not met, the graph is definitely not 2-edge connected. This is because remove any of the link, the graph much be disconnected. ■

Theorem 3: Finding an 802.16 network of minimum number of links that can achieve a given set of demands even when there is at most one non base station node failure is NP-hard.

Proof: This can be proved by reducing the minimum 2-vertex connected subgraph problem [15] to it. The reduction is very similar to the proof of Theorem 2 ■

The basic idea of both the integer programming and the greedy algorithms in this section is to not only pick a primary path but also pick a disjoint backup path for each demand such that the backup path has enough bandwidth to carry the demands when the primary path is broken due to any single link/node failure. Unlike the primary path, not all backup paths are used at the same time. Thus, some of the backup paths can share their bandwidth. The intuition behind backup bandwidth sharing can be explained as follows. In order to handle a single link/node failure, it is only necessary to reserve enough bandwidth such that only flows affected by the single link/node failure can be rerouted. Therefore, if two flows do not share any link/node in the primary path, they can share the same link(s) (and bandwidths) on the backup paths. The resource needed per link is the maximum resource required between the two flows. Based on this observation, it is possible to substantially reduce the amount of resource required for backup paths and thus potentially the number of links.

A. 0-1 Integer programming solution for MLRLP

The MLRLP can be formulated as the following integer programming problem. The basic idea is to reserve enough bandwidth on the links such that the affected flows can be rerouted using reserved bandwidth when there is a single link failure.

$$
\begin{aligned}
\min \quad & \sum_{\forall e \in E} x_e \\
\sum_{p' \in P'_v} y_{p'} \quad & \geq \quad \sum_{e \in p : p \in P_v} y_p \quad \forall e \in E, \forall v \in V \\
\sum_{v} \sum_{p \in P_v : e \in p} y_p \quad & + \quad \max_{\forall e' \neq e} \sum_{v} \sum_{e' \in p : e \in p' : p \in P_v : p' \in P'_v} y_{p'} \\
& \leq \quad x_e r(e) \quad \forall e \in E \\
\sum_{p \in P_v} y_p \quad & \geq \quad d_v \quad \forall v \\
y_p \quad & \geq \quad 0 \quad \forall p \in P_v, \forall v \\
x_e \quad & \in \quad \{0, 1\} \quad \forall e \in E
\end{aligned}
$$

Where the P_v and P'_v is the set of primary path and backup path for demand d_v. The first constraint states that there is enough bandwidth on backup paths when any single link fails. The second constraint is to ensure that the sum of the total traffic traversing a link and the capacity reserved on the link for backup paths to handle any single link failure can not exceed the link capacity if the link is in use, otherwise zero. The

MLRLP_greedy ($G=(V,E), L, D$)
1. Sort all links as
   $e \in E$ as $e_1, e_2, \ldots, e_{|E|}$ s.t. $e_i \geq e_j$ if $i < j$
2. $E_s = \{\}$, $\min = +\infty$
3. **For** $i = 1$ **to** $|E|$
4. $E_s = E_s \cup \{e_i\}$
5. If there is at least one path from s to all $v \in V$
6. R = PickLeastLinks($G' = (V, E_s), D$)
7. **If** $R \neq \{\}$
   R' = PickLeastLinksResL (G', R, D)
8. **If** $R' \neq \{\}$ and $|R'| < \min$
9. $\min = R', \min R' = R'$
10. **EndFor**
11. return $\min R'$

Fig. 6. MLRLP Greedy Algorithm

rest of the constraints is the same as in the previous integer programming for MLP.

B. 0-1 Integer programming solution for MLRNP

In the case of MLRNP, we have to reserve enough bandwidth for a single node failure. Let $A(u)$ denote the links attached to node u and $N(p)$ to denote the set of nodes on path p. The MLRNP can be formulated as the following integer programming problem, using a similar idea as in Section VI-A.

$$
\begin{aligned}
\min \quad & \sum_{\forall e \in E} x_e \\
\sum_{p' \in P'_v} y_{p'} \quad & \geq \quad \sum_{u \in N(p): p \in P_v} y_p \quad \forall u, v \in V, u \neq v \\
\sum_{v} \sum_{p \in P_v : e \in p} y_p \quad & + \quad \max_{\forall u, e \notin A(u)} \sum_{v \neq u} \sum_{p \in N(u): e \in p' : p \in P_v : p' \in P'_v} y_{p'} \\
& \leq \quad x_e r(e) \quad \forall e \in E \\
\sum_{p \in P_v} y_p \quad & \geq \quad d_v \quad \forall v \\
y_p \quad & \geq \quad 0 \quad \forall p \in P_v, \forall v \\
x_e \quad & \in \quad \{0,1\} \quad \forall e \in E
\end{aligned}
$$

Where the P_v and P'_v is the set of primary path and backup path for demand d_v. The first constraint states that there is enough bandwidth on backup paths when any single node fails. The second constraint is to ensure that the sum of the total traffic traversing a link and the capacity reserved on the link for backup paths to handle any single node failure can not exceed the link capacity if the link is in use, otherwise zero. The rest of the constraints is the same as in the previous integer programming for MLP.

C. MLRLP Greedy Algorithm

In this section, we present our MLRLP greedy algorithm. The approach is similar to the MLP presented in Section V-B and differs mainly in the selection of backup paths.

We now derive the amount of capacity reservation required when a link e is on the backup path of a set of demand $\delta = (d_{v_1}, d_{v_2}, \ldots, d_{v_k})$. Let $\theta_e^{e'}(v)$ denote the amount of capacity needed to be reserved on link e for demand d_v when link e' fails. Thus, $\theta_e^{e'}(v) = 0$ when e' is not on the primary path for demand d_v, i.e., $e' \notin p_v$; otherwise $\theta_e^{e'}(v) = d_v$. The capacity reservation on link e is $max_{\forall e' \neq e} \sum_{d_v \in \delta} \theta_e^{e'}(v)$.

PickLeastLinksResL ($G' = (V, E_s), L, D$)
1. $\forall v \in V \setminus \{s\}$
   find set of all paths in G' from s to v that are link disjoint from primary path p_v, P'_v
2. $u(e) = \sum_{v \in V} \sum_{e \in p : p \in P'_v} 1$, $\forall e \in E_s$
3. $OLD = \max_{e \in E_s} u(e)$
4. $MAX\_POPULARITY = \max_{e \in E_s} u(e) + OLD$
5. $\bar{u}(e) = MAX\_POPULARITY - u(e)$
6. $\forall e \in L$,
   $\bar{u}(e) = \bar{u}(e) - OLD$
7. Sort all $v \in V \setminus \{s\}$ as
   $v_1, v_2, \ldots, v_{|V|-1}$ s.t. $d_{v_i} \geq d_{v_j}$ if $i < j$
8. $L' = \{\}$
9. **For** $i = 1$ **to** $|V| - 1$
10. Find $p'_{v_i} = \min_{p \in P'_{v_i}} \sum_{e \in p} \bar{u}(e)$ and $\forall e \in p'_{v_i}$, $\sum_{v \in V : e \in p_v} d_v + max_{\forall e' \neq e} \sum_{v \in V} \theta_e^{e'}(v) \leq r(e)$
11. If no such p'_{v_i} can be found, return $\{\}$
12. $\forall e \in p'_{v_i} \setminus L \setminus L'$, $\bar{u}(e) = \bar{u}(e) - OLD$
13. $L' = L' \cup p_{v_i}$
14. **EndFor**
15. return L'

Fig. 7. PickLeastLinksResL

The detailed algorithm is presented in Figures 6 and 7. The MLRLP greedy algorithm, similar to the MLP greedy algorithm, sorts the links in decreasing order of their capacities (line 1 of Figure 6) and then adds one link at a time (line 4) and executes the following: a) it first places all primary paths using the greedy *PickLeastsLink(.)* (line 6) presented in the previous section. b) it then calls *PickLeastsLinkResL(.)* to find the backup paths (line 7). This process is repeated until all the links are added and then the solution of *PickLeastLinksResL(.)* that returns the least number of links added is returned.

The function *PickLeastLinksResL()* first finds all the link disjoint paths P'_v from the primary p_v for every demand d_v (line 1 of Figure 7). Each link is weighted with popularity that is the number of paths in $\cup_{\forall v \in V \setminus \{s\}} P'_v$ that traverse the link [1](line 2). Similarly as in MLP-heuristic, we convert the popularity metric, $u(.)$, to an unpopularity metric, $\bar{u}(.)$, for each link (line 5). In order to encourage the paths using less new links, all the links that are part of any primary path are awarded with an additional popularity value, OLD (line 6). For each demand, a shortest path algorithm is executed (line 10) that treats unpopularity as the link cost and finds a backup path of less hops and that traverses more shared links without violating the capacity constraint (the sum of the capacity used by all primary paths traverse the link and the reserved capacity for backup paths does not exceed the total link capacity). Once a backup path is chosen, all the links on the path receive a

[1] s represents the root node

PickLeastLinksResN $(G' = (V, E_s), L, D)$

1. $\forall v \in V \setminus \{s\}$
   find set of all paths in G' from s to v that are node disjoint from primary link p_v, P'_v
2. $u(e) = \sum_{v \in V} \sum_{e \in p : p \in P'_v} 1$, $\forall e \in E_s$
3. $OLD = \max_{e \in E_s} u(e)$
4. $MAX\_POPULARITY = \max_{e \in E_s} u(e) + OLD$
5. $\bar{u}(e) = MAX\_POPULARITY - u(e)$
6. $\forall e \in L$,
   $\bar{u}(e) = \bar{u}(e) - OLD$
7. Sort all $v \in V \setminus \{s\}$ as
   $v_1, v_2, \ldots, v_{|V|-1}$ s.t. $d_{v_i} \geq d_{v_j}$ if $i < j$
8. $L' = \{\}$
9. **For** $i = 1$ **to** $|V| - 1$
10. Find $p'_{v_i} = \min_{p \in P'_{v_i}} \sum_{e \in p} \bar{u}(e)$ and $\forall e \in p'_{v_i}$, $\sum_{v \in V : e \in p_v} d_v +$ $\max_{\forall w, e \notin A(w)} \sum_{v \neq w \in V} \theta^w_e(v) \leq r(e)$
11. If no such p'_{v_i} can be found, return $\{\}$
12. $\forall e \in p'_{v_i} \setminus L \setminus L'$, $\bar{u}(e) = \bar{u}(e) - OLD$
13. $L' = L' \cup p_{v_i}$
14. **EndFor**
15. return L'

Fig. 8. PickLeastLinksResN

popularity award OLD (line 12) through which the subsequent backup paths are encouraged to use existing links when there is enough residual capacity.

Note that there is an interesting trade-off between sharing links among primary paths and sharing bandwidth among backup paths. While in the choice of primary paths, we would like to reduce the total number of links by sharing as many links as possible, the backup paths for these primary paths that are sharing links, cannot share bandwidth. Finally, in this algorithm, we only consider flows going in a single direction and assume that the failure in one direction does not affect the failure of the other direction.

D. MLRNP greedy algorithm

The MLRNP greedy algorithm is very similar to that for MLRLP. The heuristic in Figure 6 can be used for MLRNP if we replace *PickLeastLinksResL()*with *PickLeastLinksResN()* in Figure 8. There are two major differences between *PickLeastLinksResL()* with *PickLeastLinksResN()*. First, the set of backup paths P'_v for a demand d_v using primary path p_v are the paths from root node to v that do not share any node with the primary path. Second, the bandwidth reservation required for backup paths should be valued differently. Lets assume a link e is on the backup paths of a set of paths $P = \{p_{v_1}, p_{v_2}, \ldots, p_(v_k)\}$ carrying demand $\delta = \{d_{v_1}, d_{v_2}, \ldots, d_{v_k}\}$. $\theta^u_e(v)$ denotes the backup bandwidth reservation required for demand d_v when node u fails. $A(u)$ denotes the links that are adjacent to node u. $\theta^u_e(v)$ is 0 when u is not on the primary path p_v, otherwise $\theta^u_e(v) = d_v$. The total capacity needed for backup paths on link e is $\max_{\forall u, e \notin A(u)} \sum_{p_v \in P} \theta^u_e(v)$.

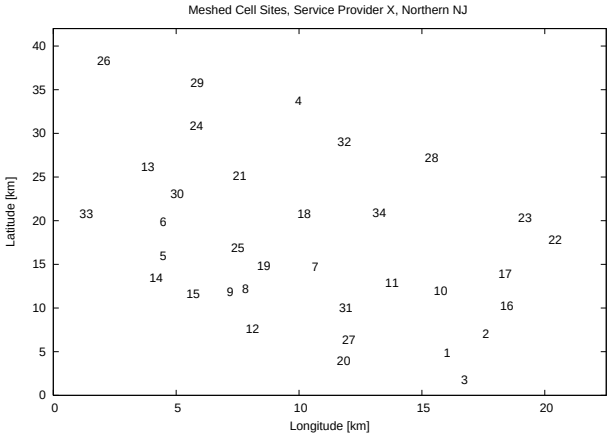


Fig. 9. Geographical locations of base stations

VII. EVALUATION

The map of base station layout that we use for the performance evaluation of our algorithms is presented in Figure 9. It is part of a 3G network operated by one of the major service providers in United States. All these base stations connect to the RNC through wireline such as T1/T3. We evaluate the number of wireless links required when we replace the wireline network with a 802.16 wireless network. The longitude and latitude distances are all relative to a reference point. Note the location of the reference point is not important. Only the distances among base stations matter in our simulation. There are a total of 34 nodes in the map. It is well known that the signal strength is inversely proportional to the square of the distance. Therefore we assume the transmission rate of a wireless links is inversely proportional to the square of the distance. We set the maximum transmission range of a wireless link to be 20Km. This gives us a total of 463 bidirectional links. The link capacities are normalized such that they range from 2.53 (longest link) to 2180 (shortest link) units. We assume the RNC is connected to one of the nodes, called the *root node*, using a wired link (such as OC3) - this allows the RNC to be located far away from the base stations, for example, in a central office. We consider the sensitivity of choosing the root node in our evaluation. The rest of the nodes are 3G base stations which demands traffic from the RNC. We assume that the demands of each base station is uniformly distributed between 0 and *Maxload*. Three different load conditions are experimented with, low load (MaxLoad=1), medium load (MaxLoad=6) and high load (MaxLoad=12.0).

All our algorithms are implemented in C. The integer programming is solved using a public domain software *lp_solve* that allows us to specify that certain variables must be integers.

A. Evaluation of MLP Greedy Algorithm

We first evaluate the *PickLeastLinks()* algorithm and show how the results are improved when using *MLP_Greedy()*. Figures 10 and 11 show how the number of links found by *PickLeastLinks()* varies with increasing size of the input graph G. First, note that the algorithm does not strictly decrease the number of links needed even though the smaller input graph is a proper subset of the larger input graph. Second, the performance can change rapidly with the addition of new

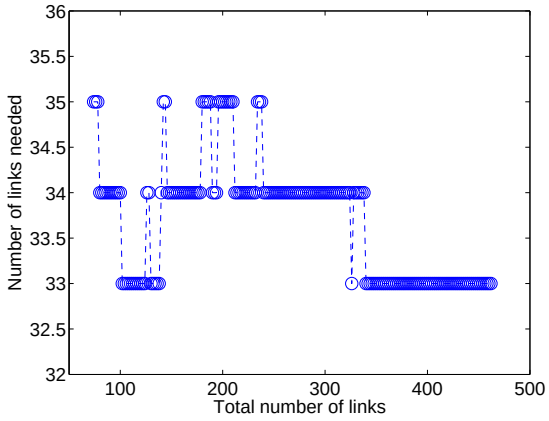


Fig. 10. Number of Links Need wrt input graph G (Medium Load, Root Node=9)

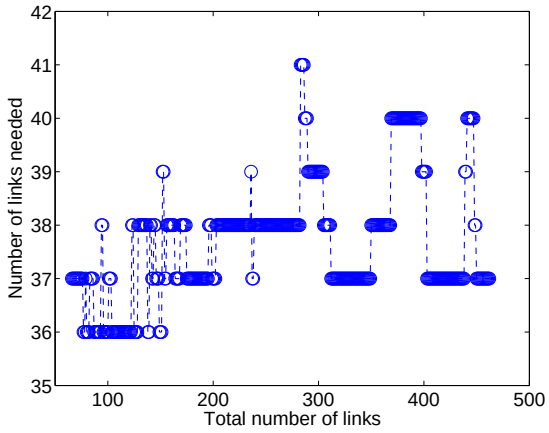


Fig. 11. Number of Links Need wrt input graph G (High Load, Root Node=9)

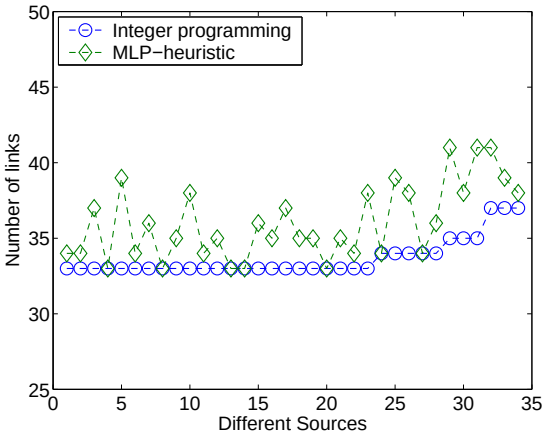


Fig. 12. Integer programming vs. MLP Greedy algorithm (small network, medium load)

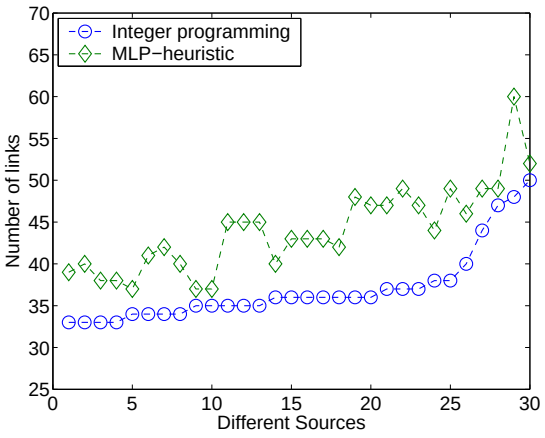


Fig. 13. Integer programming vs. MLP Greedy algorithm (small network, high load)

link(s). These observations are explained as follows. As the number of links increases, the algorithm has a larger choice of shortest paths and may end up picking the "wrong" path. Hence, a larger input graph can degrade performance. On the other hand, the addition a new link sometimes provides necessary connectivity between bottleneck nodes, resulting in a path that can be routed more efficiently with less links. Therefore, the *MLP_Greedy()* algorithm that we propose grows the input graph sequentially, link by link, and outputs the minimum number of links found by *PickLeastLinks()* among all different input graphs.

We now evaluate the performance of the *MLP Greedy* by comparing its solution with the optimal solution yielded by using integer programming. However, the integer programming can be solved in reasonable time (days) only for small size problems, i.e, a small network. As a result, we artificially reduce the original network size by reducing the maximum transmission range of each wireless link from $20km$ to $5km$ so that we can compare the solution of the MLP Greedy algorithm with the optimal solution. This is identical to removing the long range low bandwidth links from the original network and results in a network of 51 bidirectional links, that is a subset of the original network which had a total of 463 links.

Figures 12 and 13 shows the number of links obtained from both the integer programming and the *MLP Greedy* algorithm when different nodes are picked as the root node for the case of medium load and high load demands, respectively. The root nodes are sorted according to its optimal solution to make the figures easier to read. Note that the absolute minimum number of links required, without capacity constraint is to simply form a tree connecting all 34 nodes using 33 links. It can be observed for the medium load case that *MLP Greedy's performance is identical or fairly close to the optimum solution in almost all the cases*. When the load is increased, the performance gap between *MLP Greedy* and optimal widens a little, but *MLP Greedy* is still reasonably close to the optimal solution (within 2 links of optimal up to the first 13 different root nodes). It is clear from these figures that the number of links needed depends strongly on the root node selected. In fact for a few high load cases, no solution is found by *MLP Greedy* [2]. This tells us that *if there is a choice of nodes that the RNC can be connected to, one can select a "good" root node based on the minimum number of links returned by the MLP Greedy algorithm*.

Lastly, we present the results of *MLP Greedy* when the

[2]That is why there is less points in Figure 13 than in Figure 12

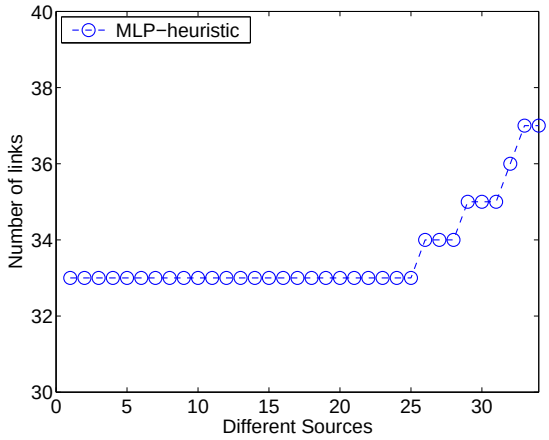


Fig. 14. MLP Greedy for large network (medium load)

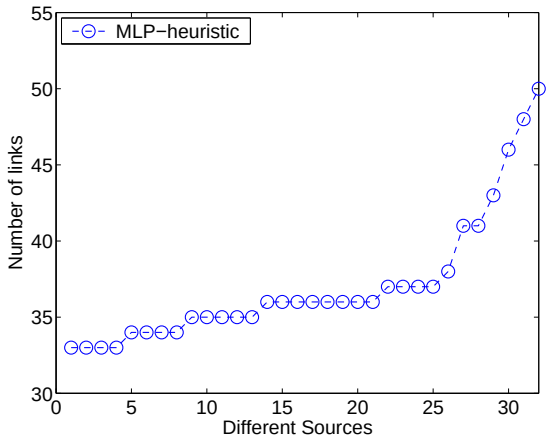


Fig. 15. MLP Greedy for large network (high load)

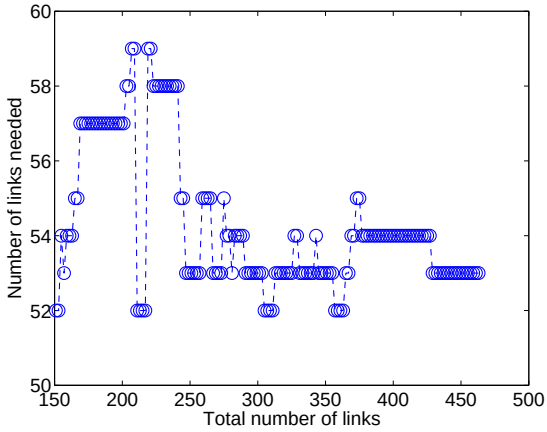


Fig. 16. Number of Links Needed wrt input graph G (Low Load, Root Node=7)

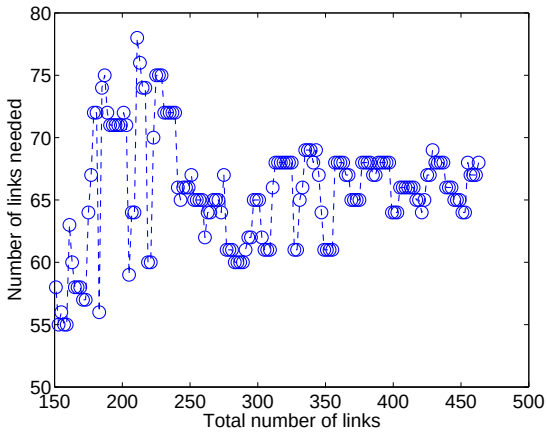


Fig. 17. Number of Links Needed wrt input graph G (Medium Load, Root Node=7)

maximum transmission range is 20km, thus, on the much larger original network, for both medium load and high load in Figures 14 and 15 respectively. Even after several days of running the integer programming solution, we obtained no solution and thus only the results for MLP Greedy are presented. Comparing the results of Figures 14 and 15 with Figures 12 and 13, we can see that the *quality of the greedy algorithm is improved when the network size is increased as there are more links to choose from*. Moreover, some infeasible problems in the smaller network case now become feasible in the original larger network. This is because the MLP Greedy algorithm tries input graphs of different sizes (sorted by the link bandwidths) and produces either the same or better solution in a large network when the small network is a subset of the large one.

B. Evaluation of MLRLP/MLRNP Greedy Algorithms

In this section, we consider the problem of designing an 802.16-based radio access network topology that is resistant to link or node failures. Since the MLRLP and MLRNP algorithms use similar heuristics, we only present the results for the MLRLP algorithm that handles link failures. The results of MLRNP algorithm are similar.

Figures 16 and 17 show how the number of links needed vary with increasing size of the input graph G for the root node 7 when *PickLeastLinkResL(.)* is used. The range of G shown is those where a solution can be found, which starts when G has more than 150 bi-directional links. For the low load case, the number of links needed is the number of links necessary to make the graph two-connected, which is 52. Compared to the case of a pure ring topology, which requires the minimum number of links, 34 to support single link failure, $18(52-34)$ additional links may appear as high overhead. However, this is not the case for our problem domain because of the following three reasons. First, typically a ring traversing all the nodes is not possible in our network given the wireless transmission range constraints. Second, a ring structure can be used only if the capacity of every link is equal or greater than the sum of all the demands - this is typically not possible over the limited bandwidth wireless links. Third, a ring structure has very long primary and backup paths which is not preferred in this problem domain since the access network delays would be excessively large.

For the case of medium load, the links needed increases from 52 to only 55. Notice that the minimum number of links required obtained for the low load and medium load cases are from very different initial graphs, G. For the low load case, the minimum number of links is found over a large range of

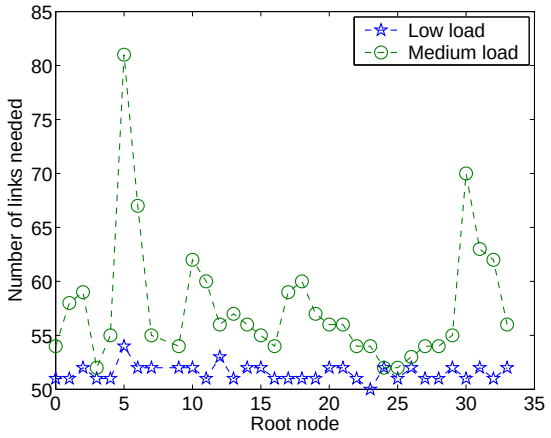


Fig. 18. Number of Links Needed wrt Root Node (Medium Load)

G but for the medium load case, the best solution occurs only when G is just large enough to obtain a solution. For the high load case, as can be expected, there is no feasible solution for accommodating the demands for the backup paths so that the topology can be made link failure resistant.

Similar to the MLP Greedy algorithm, Figures 16 and 17 suggest that the best solution obtained from *PickLeastLinkResL(.)* is when some lower bandwidth links are ignored from the input graph. This is the motivation for our *MLRLP Greedy()* which adds one link at a time to the input graph and outputs the solution of *PickLeastLinkResL(.)* with minimum links.

Finally, Figure 18 shows how the number of links needed varies with different choices of the root nodes for low and medium load cases. The case for low load is shown to serve as the base case for how many links are necessary to simply provide two connectivity with fewer capacity constraints. On average, 19.2 links are needed to provide the extra connectivity during link failures. In order to support traffic generated by the medium load case, 6.1 additional links over the low load case are required on the average.

C. Summary

We first evaluated the performance of the *MLP Greedy* algorithm for determining the topology of the radio access network with the minimum number of 802.16 links while meeting the demands of the different base stations. We showed using a smaller network (where the optimal integer programming solution is practical) that the MLP Greedy algorithm provides results that are fairly close to that of the optimal solution of 33 links. Furthermore, on the original network (where the integer programming does not result in a solution), MLP Greedy was able to improve its solution by taking advantage of availability of more paths and resulted in the optimal number of links for several choices of root nodes.

We then considered the problem of designing an 802.16-based radio access network that is resilient to link/node failures. We found that the MLRLP greedy algorithm provided feasible results for the low load and medium load cases with networks consisting of 52 and 55 links respectively. While this is significantly more than the minimum number of links necessary (34 for ring topology), given the specific constraints of the wireless backhaul in terms of limited connectivity, bandwidth etc., the two-connectivity solution provided by MLRLP greedy algorithm has a topology with significantly less than twice the number of links needed to construct a ring topology.

VIII. Conclusion

In this paper, we proposed designing an IEEE 802.16-based wireless radio access network to carry traffic in the 3G backhaul network between base stations and radio network controller. Specifically, we tackled the following question. Given a network layout (base stations and base station controllers), how should the backhaul topology be designed such that the number of 802.16 links used can be minimized while meeting the expected demands of traffic between base stations and radio network controller?

We first showed that the optimal solutions to this problem and its variants are NP-hard. The optimization problems are formulated as Integer Programming problems. We then designed the MLP greedy algorithm that performed close to the optimal solution for the case with no link failure when we evaluated the algorithm using a 3G base station layout from a major U.S. provider. Next, we considered the issue of link or node failure in the 802.16 network and designed algorithms (MLRLP/MLRNP Greedy) to find topologies that can handle single failure effectively. Using simulation, we found that the MLRLP greedy algorithm was able to design a topology with significantly less than twice the number of links needed in constructing a ring topology (if it is feasible), while handling the demands subjected to possible single link failure.

References

[1] http://www.cdg.org, "Cdma development group."
[2] C. Eklund, R. Marks, K. Stanwood, and S. Wang, "Ieee standard 802.16: A technical overview of the wireless man air interface for the broadband wireless access," in *IEEE Communications Magazine*, June 2002.
[3] http://www.wimaxforum.org, "Wimax forum."
[4] L. Bos and S. Leroy, "Toward an all-ip-based umts system architecture," *IEEE Network*, January 2001.
[5] G. Heijenk, G. Karagiannis, V. Rexhepi, and L. Westberg, "Diffserv resource management in ip-based radio access networks," in *Proceedings of 4th International Symposium on Wireless Personal Multimedia Communications (WPMC'01)*, (Aalborg, Denmark), September 2001.
[6] M. W. I. Forum, "Mtr.006v2, ip as transport in the ran," April 2001.
[7] S. Kasera, R. Ramjee, S. Thuel, and X. Wang, "Congestion control policies for ip-based cdma radio access networks," in *Proceedings of Infocom*, (San Francisco, CA), April 2003.
[8] H. el Allali and G. Heijenk, "Resource management in ip-based radio access networks," in *Proceedings CTIT Workshop on Mobile Communications*, February 2001.
[9] T. Bu, M. Chan, and R. Ramjee, "Connectivity, resilience and performance of ip-based radio access networks," in *IEEE Infocom*, March 2004.
[10] R. Jothi and B. Raghavachari, "Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design," in *Proceedings of ICALP*, April 2004.
[11] F. Salman, J. Cheriyan, R. Ravi, and S. Subramanian, "Buy-at-bulk network design: Approximating the single-sink edge installation problem," in *Proceedings of 8th ACM-SIAM Symp. on Discrete Algorithms*, 1997.
[12] R. Jothi and B. Raghavachari, "Survivable network design: The capacitated minimum spanning network problem," *Information Processing Letters*, 2004.
[13] http://www db.stanford.edu/pleiades/SUMATRA.html, "Stanford university mobility traces (sumatra)."
[14] D. Tang and M. Baker, "Analysis of a metropolitan-area wireless network," in *Proceedings of ACM Mobicom*, 1999.
[15] http://www.nada.kth.se/ viggo/problemlist/compendium.html, "A compendium of np optimization problems."