Relaxed Tree Search MIMO Signal Detection Algorithm Design and VLSI Implementation

Sizhong Chen*,Tong Zhang*, and Manish Goel†
* ECSE Department, Rensselaer Polytechnic Institute, Troy, NY
† Texas Instruments, Dallas, TX

***Abstract*— This paper presents an implementation-oriented breadth-first tree search MIMO detector design solution. Techniques at algorithm and VLSI architecture levels are developed to improve the implementation efficiency. Using Synopsys synthesis tool with 0.18μm CMOS technology, we designed soft-output detectors for 4×4 MIMO channel with 64-QAM modulation, which can achieve close to 30 Mbps throughput. To the best of our knowledge, the presented detectors are the first one reported in the open literature capable of supporting tree search soft-output MIMO detection for 4×4 MIMO transmission with 64-QAM modulation.**

I. Introduction

Signal detector is a key element in multiple-input multiple-output (MIMO) communication devices. In general, optimal maximum-likelihood (ML) hard-output and maximum *a posteriori* (MAP) soft-output MIMO detectors incur prohibitive computational complexity due to their essence of exhaustive search. To reduce the computational complexity while maintaining (near-)optimal performance, researchers have developed several nonlinear detectors that perform *non-exhaustive tree search* based on a set of additive metrics. Depending on how to carry out the non-exhaustive tree search, existing nonlinear detectors fall into three categories, i.e., depth-first search, metric-first search, and breadth-first search. VLSI architecture design and implementations of depth-first and breadth-first detectors have been addressed in [1]–[4], which nevertheless can only support up-to 16-QAM modulation for a moderate-size MIMO channel such as 4×4. In order to support higher order modulation such as 64-QAM, which will dramatically increase the computational complexity compared with 16-QAM, the detector should have sufficient operational parallelism for realizing reasonably high detection throughput. The serial nature of conventional depth-first search algorithm itself and the sorting operation in conventional breadth-first search algorithm make them incapable of effectively supporting 64-QAM modulation. This leaves the nonlinear tree search detectors that can support 64-QAM modulation for a moderate-size MIMO channel missing in the open literature.

Attempting to fill this gap, this paper presents algorithm and VLSI architecture techniques to design MIMO detectors that perform breadth-first tree search and can support moderate-size MIMO channel with high order modulation such as 4×4 MIMO channel with 64-QAM modulation. The basic idea is to improve the operational parallelism in breadth-first search by replacing the original strict sorting operation with distributed and approximate sorting operations. We also developed a technique to further reduce the computational complexity for high order modulation such as 64-QAM. VLSI architectures have been developed to implement the proposed modified breadth-first detection algorithm, and the resulted detectors are referred to as relaxed *K*-best detectors. For the purpose of demonstration, we designed two soft-output relaxed *K*-best detectors, with different silicon area vs. detection performance tradeoffs, for 4×4 MIMO channel with 64-QAM modulation. Simulation results show that the detectors, when being concatenated with a low-density parity-check (LDPC) code, can achieve the performance very close to the one using an exhaustive-search based sphere detection scheme. Synopsys tool was used to synthesize these two detectors with 0.18μm CMOS standard cell and memory libraries. With the silicon area of 10.5mm^2 and 16.3mm^2, these two detectors can achieve close to 30Mbps detection throughput when the block error rate is around 10^{-3}.

II. Background

A. MIMO Signal Detection

This work considers the MIMO system with *spatial multiplexing* signaling. Let N_t and N_r represent the number of transmit and receive antennas, respectively. The MIMO transmission can be modelled as $\mathbf{y} = \mathbf{H}\cdot\mathbf{s}+\mathbf{n}$, where $\mathbf{y}$ is a signal vector received by a MIMO detector, $\mathbf{H}$ is an $N_r \times N_t$ channel matrix, $\mathbf{s}$ is a transmitted symbol vector, and $\mathbf{n}$ is a Gaussian noise vector with variance of $N_0/2$. Let $\mathbf{x}$ denote the transmitted binary vector, we have $\mathbf{s}$=map($\mathbf{x}$). Following the principle of maximum likelihood (ML) detection, the task of a hard-output detector is to solve

$$\min_{\mathbf{s}\in\Omega} \|\mathbf{y} - \mathbf{H}\cdot\mathbf{s}\|^2, \tag{1}$$

where Ω contains all the possible transmitted symbol vectors. The task of a soft-output detector is to compute the log-likelihood value of each bit, which is defined as $L(x_i|\mathbf{y}) = \ln\frac{P(x_i=+1|\mathbf{y})}{P(x_i=-1|\mathbf{y})}$. Through standard simplification [5], [6], $L(x_i|\mathbf{y})$ can be approximated as:

$$\begin{aligned} &L(x_i|\mathbf{y}) \approx \max_{x_i=+1}\{\Lambda(\mathbf{x},\mathbf{y}\} - \max_{x_i=-1}\{\Lambda(\mathbf{x},\mathbf{y})\}, \\ &\text{where} \quad \Lambda(\mathbf{x},\mathbf{y}) = -\frac{1}{N_0}\|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2. \end{aligned} \tag{2}$$

As discussed in the literature (e.g., see [5], [6]), we can rewrite (1) and $\Lambda(\mathbf{x},\mathbf{y})$ in (2) as

$$\min_{\mathbf{s}\in\Omega}\Big(\sum_{i=1}^{N_t}\Big|\sum_{j=1}^{i} l_{i,j}(s_j - \hat{s}_j)\Big|^2\Big) = \min_{\mathbf{s}\in\Omega}\Big(\sum_{i=1}^{N_t}\Lambda_i^h\Big) \tag{3}$$

and

$$\Lambda(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N_t}\Big(-\frac{1}{N_0}\Big|\sum_{j=1}^{i} l_{i,j}(s_j-\hat{s}_j)\Big|^2\Big) = \sum_{i=1}^{N_t}\Lambda_i^s. \tag{4}$$

where $\mathbf{L} = (l_{i,j})$ is a lower triangular matrix obtained from $\mathbf{H}$ by standard matrix decompositions such as Cholesky or QR decomposition and $\hat{\mathbf{s}} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\mathbf{y}$. Hence, we obtain *additive metrics* with the metric increments Λ_i^h and Λ_i^s that depend only on s_j for $j \le i$. Since the term $-\frac{1}{N_0}$ in (4) can be omitted in the tree search, the metric increment Λ_i^s for soft-output detection becomes equivalent to the metric increment Λ_i^h for hard-output detection. Therefore, we simply denote the metric increment as Λ_i and define the metric of a depth-n path as $\Gamma^{(n)} = \sum_{i=1}^{n}\Lambda_i$.

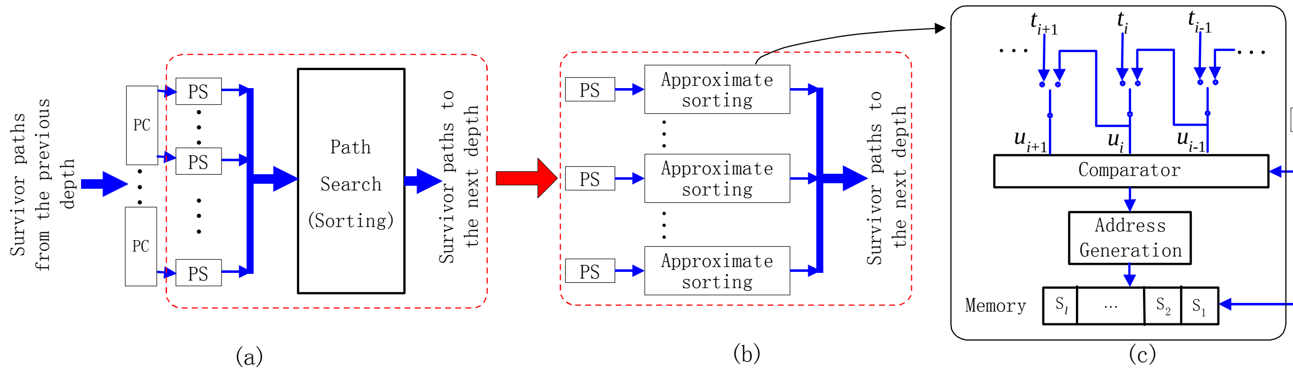

Fig. 1. (a) Structure of the processing unit at each depth in a *K*-best detector, (b) structure of the distributed and approximate sorting, and (c) realization of the approximate sorting.

B. Breadth-First Search K-Best Detector

From the above discussion, we know that hard/soft-output detection can be formulated as tree search problems based on the additive metric Λ_i. Applying the principle of *M*-algorithm [7] to perform breadth-first tree search MIMO detection, we have the so-called *K*-best detector [3], [4]. A *K*-best detector performs the following operations at depth d:

1) *Path Extension*: Extend each survivor path from the previous depth, i.e., calculate $\mathit{\Gamma}^{(d)} = \mathit{\Gamma}^{(d-1)} + \Lambda_d$ for each modulation point.
2) *Radius Check*: Delete the extended paths whose metrics are larger than a pre-defined value r that is equivalent to the radius in the sphere detection algorithm.
3) *Path Search*: Let *R* denote the number of the remaining extended paths. Sort the *R* extended paths in ascending order based on the path metric and select the first $\min(R, K)$ paths as survivors at depth d.

Hard- and soft- output *K*-best detectors only differ on how to generate the output using the survivors obtained after reaching the tree leaves. In general, to provide high quality soft-output information and ensure good performance, a soft-output detector typically requires a (much) larger value of *K* than that of its hard-output counterpart. Thus a soft-output detector usually has much higher computational complexity and hence silicon overhead.

III. Relaxed *K*-Best Detector

In this section, we first elaborate on two inherent drawbacks of the original *K*-best detector in terms of VLSI implementation, then present our solutions to tackle these issues, which will lead to so-called relaxed *K*-best detectors.

From Section II-A, we have that the calculation of metric increments at depth-i can be written as $|P_c - P_s|^2$, where $P_c = \sum_{j=1}^{i-1} l_{i,j}(\hat{s}_j - s_j) + l_{i,i}\hat{s}_i$ that is common to all the paths extended from the same survivor, and $P_s = l_{i,i}s_i$ that corresponds to each modulation point. Therefore, to extend one survivor, we need to calculate only one P_c but multiple $|P_c - P_s|^2$. Straightforwardly, we have the generic structure of the processing unit at each depth as shown in Fig. 1(a). Each PC block calculates the P_c, and each PS block calculates the metrics, i.e., $\mathit{\Gamma}^{(i-1)} + |P_c - P_s|^2$, of all the paths extended from the same survivor and deletes those that fail the radius check. Due to the computational complexity mismatch between PC and PS blocks, several PS blocks can share one PC block. The output of all the PS blocks, i.e., the extended paths that pass the radius check, are sent to a sorting block that selects the best *K* extended paths as survivors. Such straightforward structure has two critical drawbacks, particularly for high order modulation such as 64-QAM:

1) The detector explicitly examines the extension of each survivor with all the modulation points. Due to the complex computation involved in each path extension, this will incur a large computational complexity overhead.
2) Due to its serial nature, the sorting at each depth will incur a large delay and hence become an essential throughput bottleneck. This fails to match the inherent parallelism within the path extension and radius check. Besides the large delay, sorting will also incur large silicon cost and a large amount of data movement, which will directly lead to high power consumption.

A. Improved PSK enumeration

How to tackle the first issue above has been addressed in the context of depth-first tree search detection [1], [5]. The technique is called PSK enumeration, where the basic idea is described as follows: For QAM modulation, all the modulation points locate on several circles concentric with the origin, e.g., there are 1, 3, and 9 concentric circles in QPSK, 16-QAM, and 64-QAM, respectively. It can be proved that all the points on the same circle that satisfy the radius check are always adjacent and hence form a single admissible region along the circle. By identifying the boundary of the admissible region on each circle, we do not need to explicitly examine the points outside the admissible region, leading to a significant saving of computational complexity.

However, using the original PSK enumeration method, we have to explicitly examine all the concentric circles. Denote the circles that contain modulation points satisfying the radius check as valid circles. We observe that not all the cocentric circles may be valid circles, and it is desirable if we only examine those valid circles instead of all the circles, particularly for higher order modulations. From the discussion on PSK enumeration in [5], it can be readily derived that all the valid circles fall into a continuous region. We can identify the boundary of the region containing all the valid circles as follows: For a modulation point to survive the radius check, the metric increment at depth-i must satisfy $|P_c - l_{i,i}s_i|^2 < r - \mathit{\Gamma}^{(i-1)}$, which can be reformulated as $|\frac{P_c}{l_{i,i}} - s_i|^2 < \frac{r-\mathit{\Gamma}^{(i-1)}}{l_{i,i}^2}$. We can easily find the circle closest to the point $\frac{P_c}{l_{i,i}}$ on each side, as shown in Fig.2. Extending from these two circles with the distance of $r_e = \frac{r-\mathit{\Gamma}^{(i-1)}}{l_{i,i}^2}$ on both inward and outward directions, we obtain the boundary of the valid circle region. Any circle that falls outside does not contain any modulation points that may satisfy the radius check, hence can be simply excluded from explicit PSK enumeration. Since the distance between any PSK circles is available beforehand, the boundary can be easily determined. Moreover, we note that we can pre-compute each $1/l_{i,i}$ after estimating the MIMO channel matrix, hence the computation here involves multiplication instead of division.

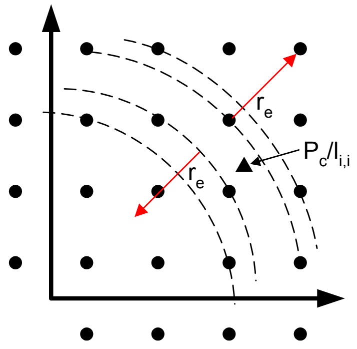

Fig. 2. Identification of valid circles.

B. Distributed and Approximate Sorting

To tackle the second issue above, we propose to replace the original strict sorting with a memory based distributed and approximate sorting. As illustrated in Fig.1(b), the basic idea is described as follows: Instead of sharing one big sorter among all the PS blocks as shown in Fig.1(a), each PS block has its own sorter and all the sorters perform approximate sorting independent from each other. The principle of approximate sorting is explained as follows: We divide the entire range of the path metric into a certain number of regions, based on which we arrange the paths extended by the same PS block into groups. Within each group, the extended paths are not sorted at all. Clearly, such approximate sorting only involves the comparison with fixed threshold values, which can be directly implemented in parallel.

Intuitively, we can use a memory block to implement such approximate sorting as follows: We uniformly partition the memory address space into l consecutive segments. Since all the incoming paths have the metric better (i.e., less) than the radius r that is used in the radius check, we choose $l+1$ threshold values $t_0 = 0 < t_1 < t_2 < \cdots < t_l = r$, and assign the range $(t_{i-1}, t_i]$ to segment S_i for $i = 1, 2, \cdots, l$. We have $t_i = i \cdot t_l/l$ for $i = 1, 2, \cdots, l-1$. Each path, whose metric falls into the range $(t_{i-1}, t_i]$, is simply stored into the memory segment S_i. Each segment has one counter to hold the address of the next available memory location.

In case that one memory segment becomes full, if we simply throw away any further incoming path to that segment, our simulations suggest that it will incur significant performance degradation. Thus we propose to make the threshold range associated with each memory segment configurable, as illustrated in Fig.1(c). Let $(u_{i-1}, u_i]$ represent the configurable threshold range associated with segment S_i. Initially, we set $u_i = t_i$ for $i = 1, 2, \cdots, l$. Before the segment S_i becomes full, we have $u_i = t_i$. Once S_i becomes full, by using a switch as shown in Fig.1(c) we configure $u_i = u_{i-1}$ so that the lower bound of the threshold range of the next segment will automatically extend from t_i (i.e., the previous value of u_i) to u_{i-1}. In this way, the range of S_i is handed over to the next (with higher value of index) segment.

The survivor paths at each depth are fed to the next depth as follows: We start with fetching one path in segment S_1 as survivor from each memory block at one time, alternatively among all the memory blocks, until we have fetched K paths or all the path stored in all the S_1 segments have been fetched. If latter happens, we move on to the segment S_2, and so on. From the hardware implementation standpoint, this memory based distributed and approximate sorting has the following main advantages:

1) The computational complexity is much less than the strict sorting, leading to a great potential of higher throughput and significant power savings;
2) There is no data movement at all as in the strict sorting, hence further reduce the power consumption;
3) The distributed structure well matches the parallelism in the path extension and hence helps to realize high throughput.

It can be intuitively justified that a bigger value of l and/or a larger size of memory will improve the detection performance at the cost of larger silicon area. In practice, we have to rely on extensive simulations to choose the appropriate values of l and total memory size subject to the desired silicon area vs. detection performance trade-offs.

C. Overall Detector Structure

Fig. 3 shows the structure of an unrolled relaxed K-best detector, which is optimized for detection throughput by mapping different tree search depth onto different depth processing unit (DPU). The path extension at each depth contains several PC blocks, and each PC block is shared by several PS blocks. The ratio between the numbers of PC and PS blocks is dependent on the target modulation size. Extending one survivor path at a time, each PS block consists of one or more PSK units, where each PSK unit non-exhaustively examines the modulation points on the same circle once a time. Since one PS block connects one approximate sorting block and all the PSK units may generate extended path that passes the radius check every clock cycle, the number of PSK units in each PS block is limited by the ratio between the speed of approximate sorting and the speed of PSK unit.

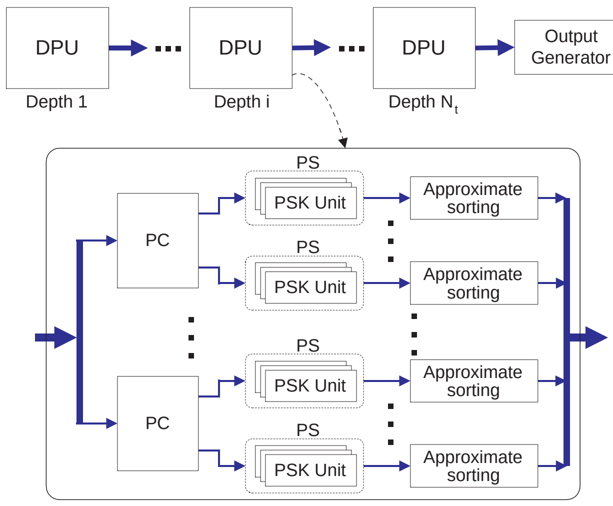

Fig. 3. Unrolled relaxed K-best detector structure.

As shown in Fig.3, an output generator is used to generate the detector output based on the last-depth survivors. For hard-output detection, it simply searches the survivor path with the best metric and output the corresponding hard decisions. For soft-output detection, we need to evaluate the L-value of each bit according to (2) based on all the last-depth survivors. We note that it is possible that all the survivor paths agree on one bit position, i.e., one of the two terms in (2) remains as undefined. As a result, the L-value of this bit cannot be directly calculated. To solve this problem, in this work, we use the worst metric among all the K final survivor paths to replace all the undefined terms. Finally, we note that the detector does not include the matrix decomposition functional block that computes the lower triangular matrix $\mathbf{L}$. Readers are referred to [8] for the discussion of implementing such a functional block.

IV. Design Example

We designed two soft-output relaxed K-best detectors, with K of 64 and 128, respectively, for 4×4 MIMO transmission with 64-QAM modulation. The decoders contain 8 and 16 PS blocks at each depth for K=64 and K = 128, respectively. At each depth, one PC block is deeply pipelined to feed all the PS blocks. Each PS block consists of 2 PSK units. Each approximate sorter contains two single-port memory blocks that receive the data from the current depth and provide the data to the next depth, alternatively. Each single-port memory can store totally 128 path data and is partitioned into 16 segments.

To evaluate the detection performance, we consider MIMO-OFDM transmission, where OFDM is based on 64-point FFT as in the IEEE 802.11a standard. Each sub-carrier MIMO channel is flat fading, i.e., all the entries in the MIMO channel matrix are independent random Gaussian variables. In the simulation, the soft-output is fed to a length 2048, rate-1/2 LDPC code decoder, which performs up to 20 decoding iterations. For the purpose of comparison, we designed a soft-output MAP detector subject to the sphere constraint, i.e., the detector exhaustively examines the paths that satisfy the sphere radius check to obtain the soft-output. The radius is the same as the one used in the relaxed K-best detectors. Fig.4 shows the simulated block error rate performance, where the relaxed K-best detectors only incur very small performance degradation. We note that, as proposed in [5], the radius r is calculated as $2\alpha N_r \sigma^2$, where α is a predefined constant parameter and σ is the noise standard deviation.

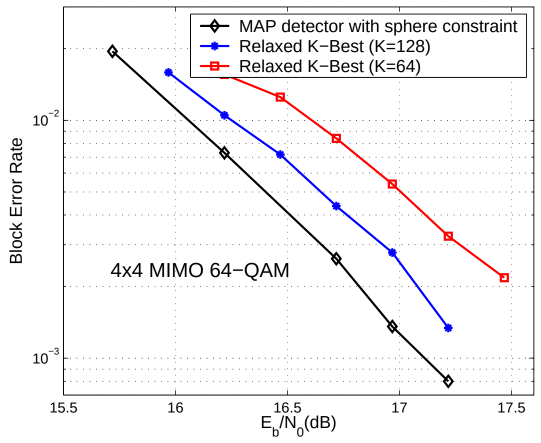

Fig. 4. Simulation results for 4×4 64-QAM.

The above relaxed K-best detectors have been designed with Verilog and synthesized using Synopsys tools with 0.18μm standard cell and memory libraries. The detectors operate with two synchronized clock signals: each PSK unit operates with a clock frequency of 166.5MHz, all the other functional blocks including the PC blocks and the approximate sorters operate with a clock frequency of 333MHz. The two detectors have the core areas of 10.5mm^2 (K=64) and 16.3mm^2 (K = 128), respectively. The detection throughput depends on the average number of survivors generated at each depth that largely depends on the run-time environment such as average SNR and instantaneous channel gain. Table I shows the average throughput of the two detectors at various SNR over the flat fading MIMO channel. Finally, we note that these detectors can also be configured to support the soft-output detection for 16-QAM and QPSK and hard-output detection for 64-QAM, 16-QAM, and QPSK, where much higher throughput can be realized mainly because of the much less average number of survivor paths, particularly in hard-output detection.

TABLE I
Throughput at various SNR of the two detectors.

E_b/N_0	17.5dB	17.0dB	16.5dB
K=128	26.9Mbps	25.2Mbps	23.6Mbps
K=64	28.2Mbps	26.0Mbps	24.0Mbps

V. Conclusions

This paper presents a nonlinear breadth-first tree search MIMO signal detector hardware design solution that supports 4×4 MIMO transmission with 64-QAM modulation. Algorithm and VLSI architecture level techniques have been developed to improve the detector implementation efficiency, while maintaining good detection performance. Proof-of-concept hardware prototypes of soft-output detectors that support 4×4 MIMO transmission with 64-QAM have been designed using 0.18μm CMOS technology. With the silicon area of less than 17mm^2, the detectors can achieve the throughput close to 30Mbps.

References

[1] A. Burg et al., "VLSI implementation of MIMO detection using the sphere decoding algorithm," *Journal of Solid-State Circuits*, vol. 40, pp. 1566 – 1577, July 2005.

[2] D. Garrett et al., "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1544–1552, Sept. 2004.

[3] K.-W. Wong, C.-Y. Tsui, R. S. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *IEEE International Symposium on Circuits and Systems*, May 2002, pp. III–273–III–276.

[4] Z. Guo and P. Nilsson, "VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proc. of IEEE CAS Symposium on Emerging Technologies*, 2004, pp. 65–68.

[5] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389–399, March 2003.

[6] S. Baro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. of IEEE International Conference on Communications*, May 2003, pp. 2653–2657.

[7] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Transactions on Communications*, vol. 32, pp. 169–176, Feb. 1984.

[8] L.M. Davis, "Scaled and decoupled Cholesky and QR decompositions with application to spherical MIMO detection," in *Proc. of IEEE Wireless Communications and Networking*, March 2003, pp. 326 – 331.