

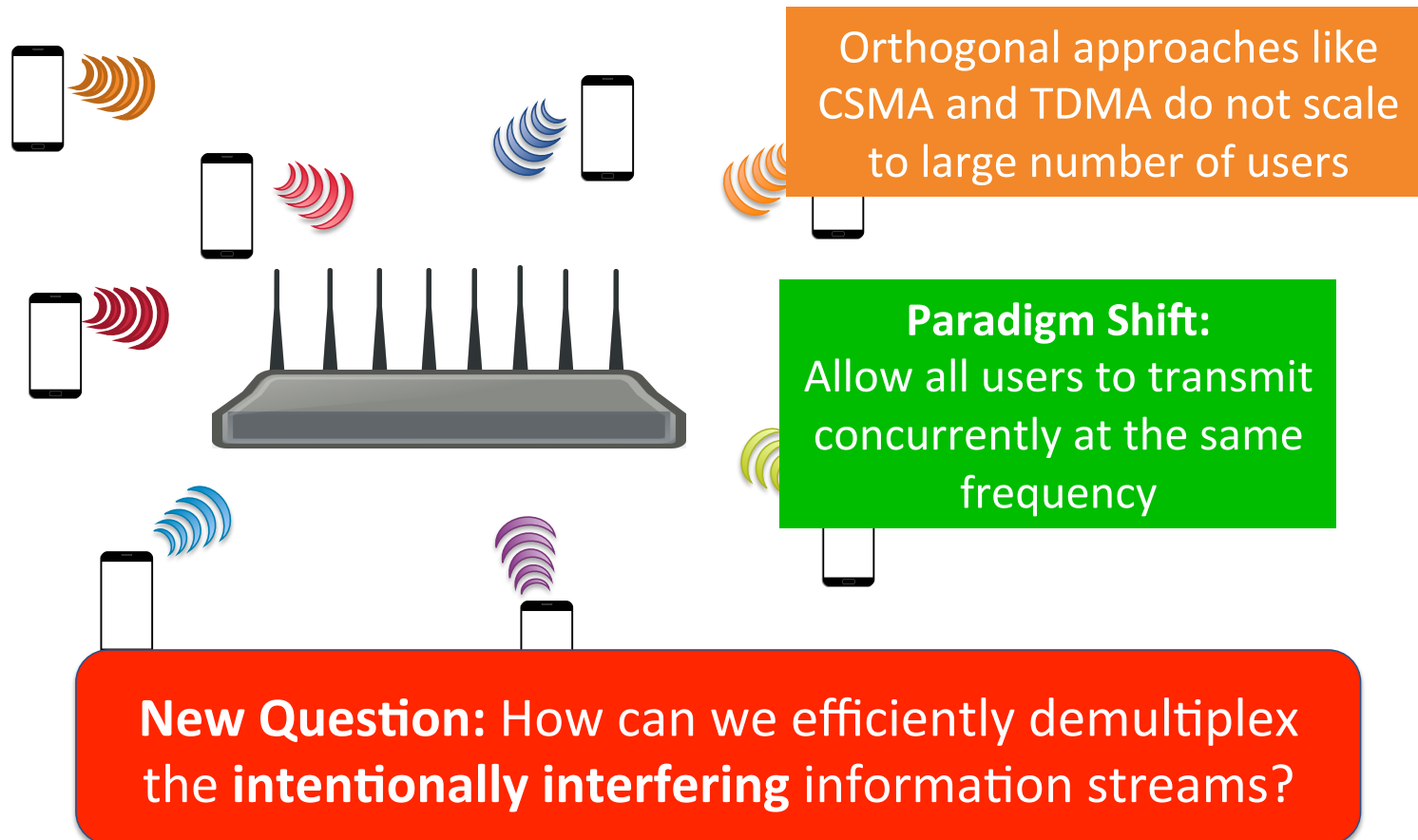
FlexCore: Massively Parallel and Flexible Processing for Large MIMO Access Points

Konstantinos Nikitopoulos⁺

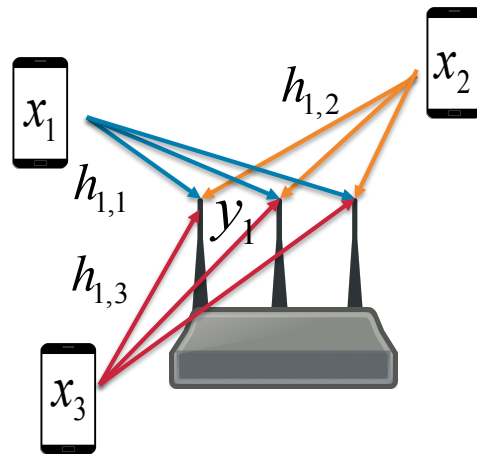
Joint work with Christopher Husmann⁺, Georgios Georgis⁺,
and Kyle Jamieson^{*,**}



From Orthogonal Transmissions to Mutually Interfering Transmissions



From Orthogonal to Mutually Interfering Transmissions



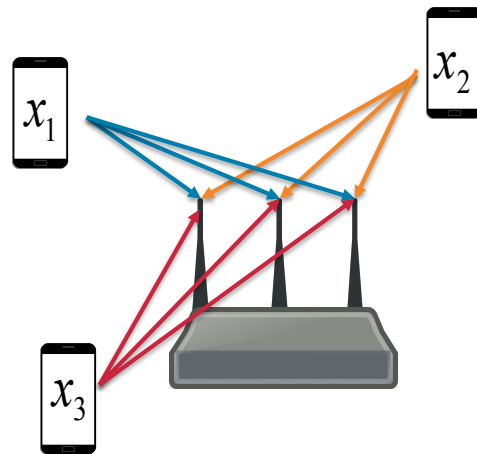
$$\underbrace{\begin{bmatrix} y_1 \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} \end{pmatrix}}_{\mathbf{H}} \cdot \underbrace{\begin{bmatrix} x_1 \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} n_1 \end{bmatrix}}_{\text{noise}}$$

- **Linear Detectors** (e.g., ZF, MMSE) offer low latency and complexity but can result in **highly suboptimal throughput**
- **Maximum-Likelihood (ML) Detection** maximizes throughput but is **highly complex**

$$\hat{\mathbf{x}} = \arg \min_{\text{possible } \mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \approx 4.2 \times 10^9 \text{ possibilities for a 16-QAM } 8 \times 8 \text{ (8 clients, 8 AP antennas)}$$


ML detection is a **fundamental problem**, with special cases solved (e.g., in channel decoding)

Sphere Decoding

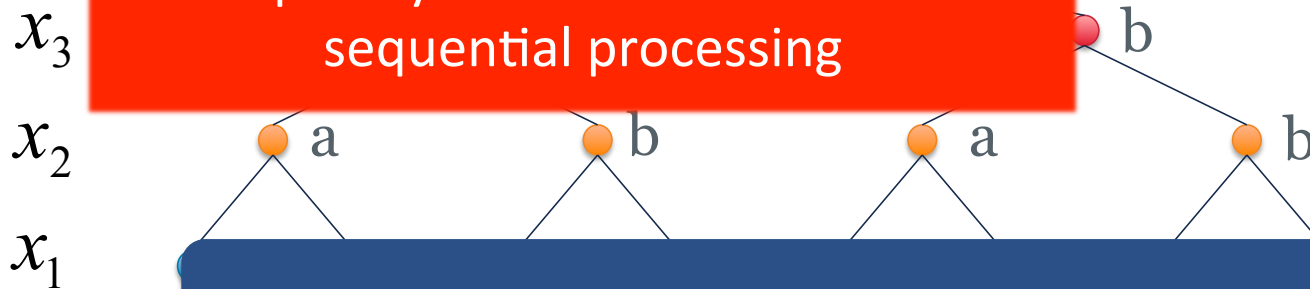


A **Sphere Decoder** transforms the exhaustive search into a tree search:

$$\hat{\mathbf{x}} = \arg \min_{\text{possible } \mathbf{x}} \|\mathbf{y}' - \mathbf{R}\mathbf{x}\|^2$$

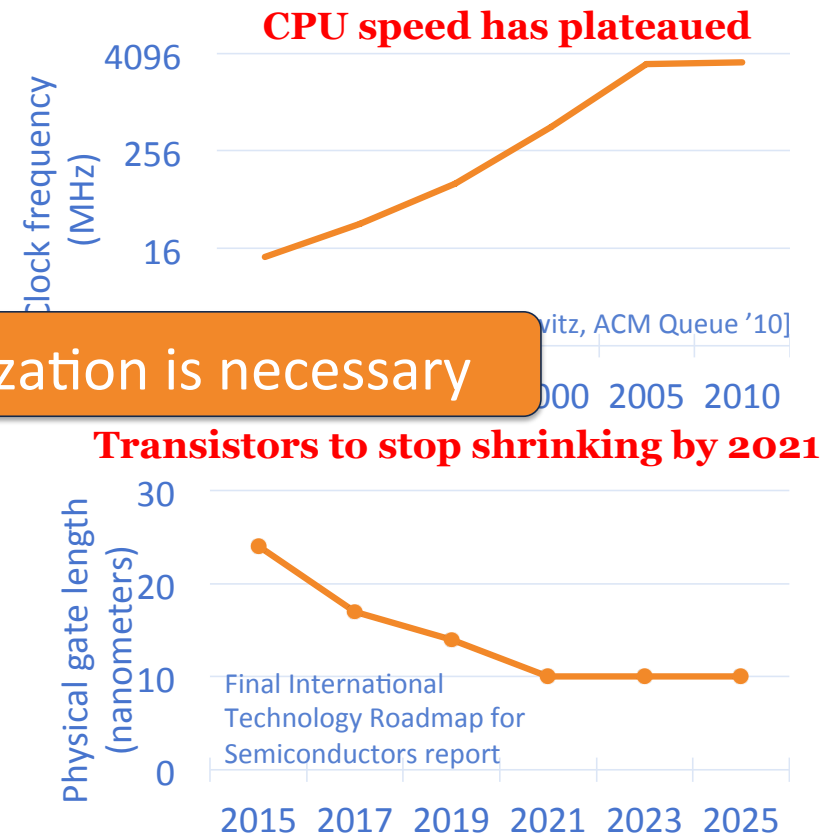
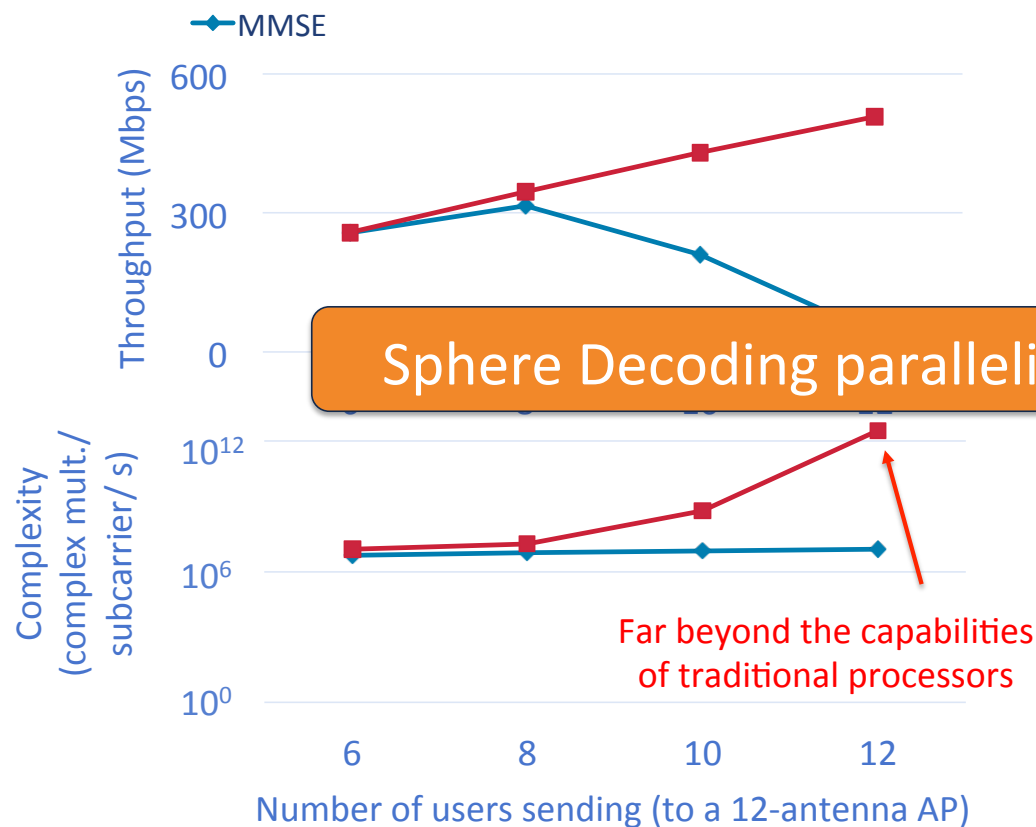
Each client i sends $x_i \in$ 

Complexity efficient tree search →
sequential processing



Sphere Decoding Goal: Find the distance-minimizing leaf

The Need for Parallelization

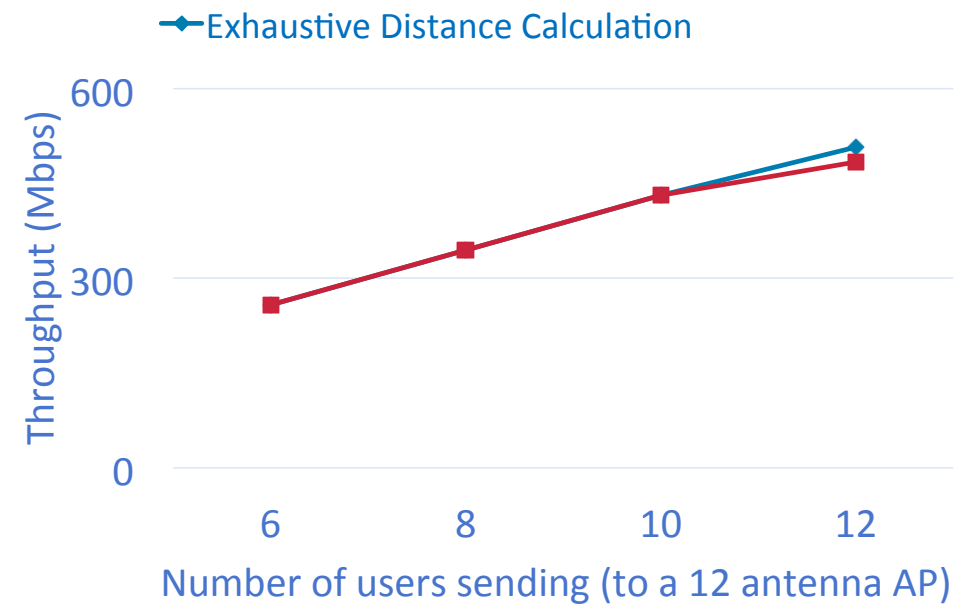
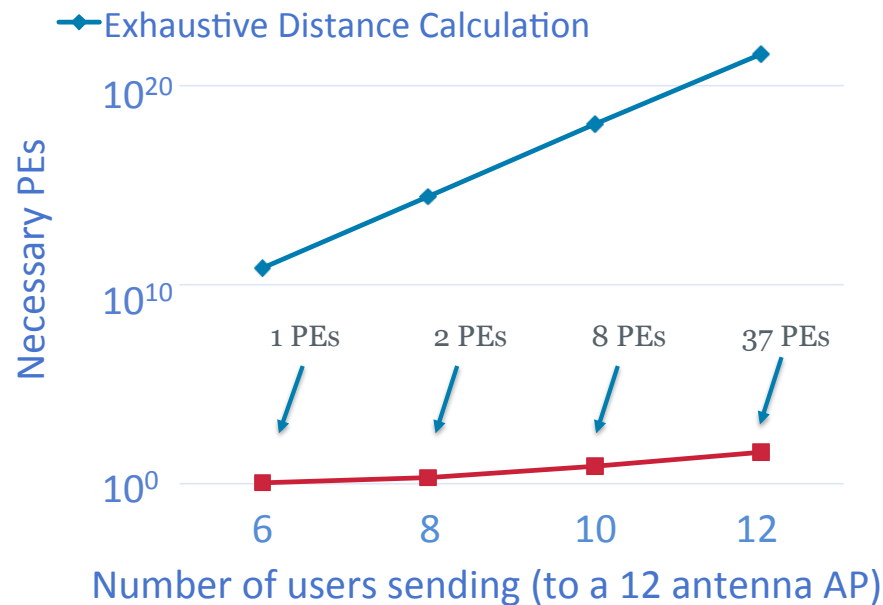


Simulation on channel traces at 21.5 dB; 20MHZ (256 subcarriers); 64-QAM

A Parallelization Attempt

$$\hat{\mathbf{x}} = \arg \min_{\text{possible } \mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$$

We could achieve minimum latency by performing the Euclidean distance calculations for each \mathbf{x} on a separate PE in parallel.



Simulation on channel traces at 21.5 dB; 20MHZ (256 subcarriers); 64-QAM

FlexCore's New Approach

Let's revisit the maximum-likelihood detection problem:

$$\hat{\mathbf{x}} = \arg \min_{\text{possible } \mathbf{x}} \left\| \overset{\text{✗}}{\mathbf{y}} - \overset{\text{✓}}{\mathbf{H}}\mathbf{x} \right\|^2$$

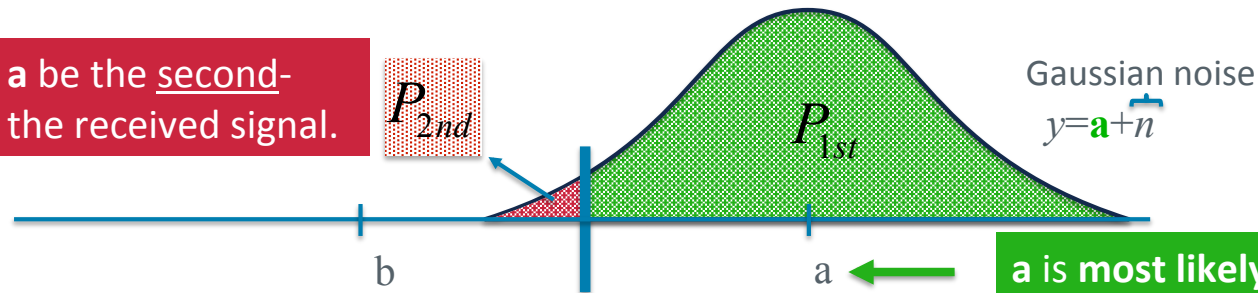
Can we identify the most likely data \mathbf{x} the users sent:

1. **Before** we receive any signal \mathbf{y} ?
2. **Only by looking** at the channel matrix \mathbf{H} ?

Primer: Detection and Probability

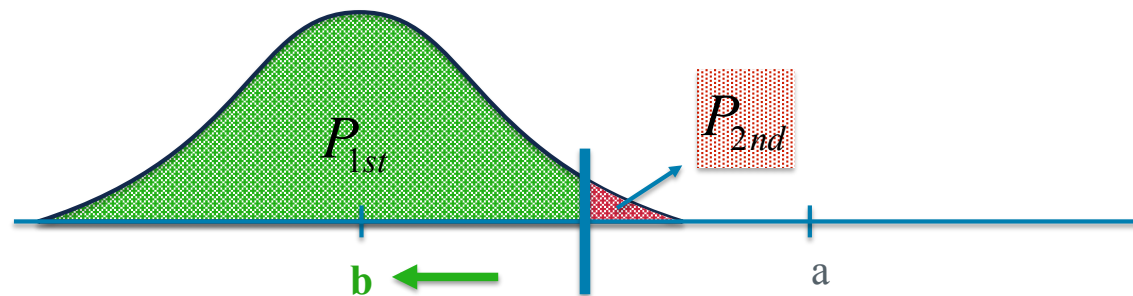
Assume the transmitted symbol is **a**:

It is **less likely** for **a** be the second-closest symbol to the received signal.



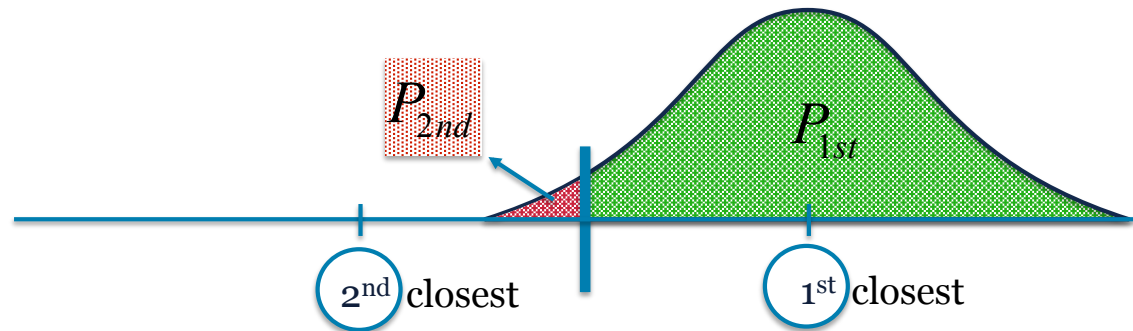
a is **most likely** to be the first-closest symbol to the received signal.

Assume the transmitted symbol is **b**:



From Absolute to Relative Probabilities

Assume the transmitted symbol is **a**:



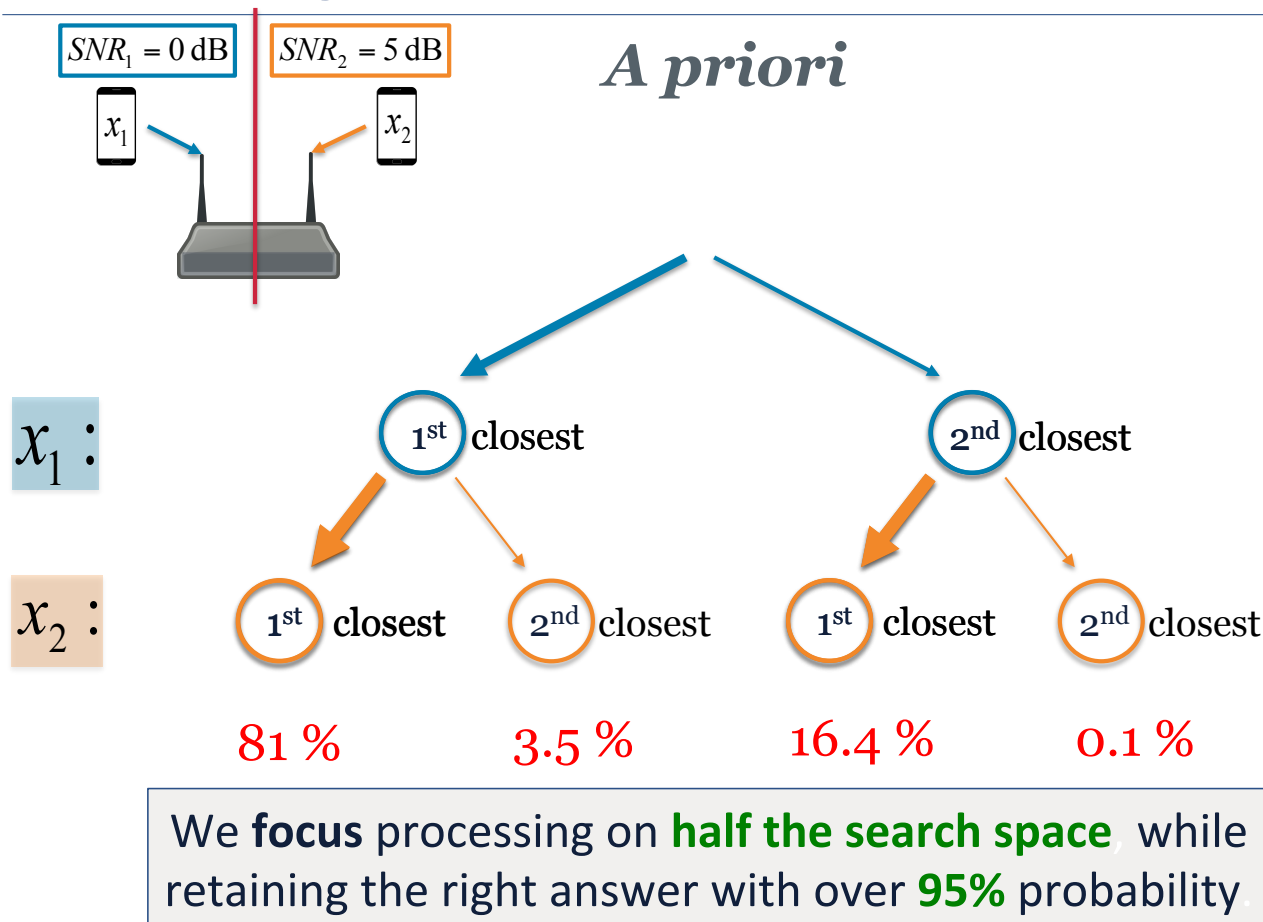
Assume the transmitted symbol is **b**:

Any transmitted symbol is **more likely** to be the **first closest** to the received signal, and **less likely** to be the **second closest**

The corresponding probabilities can be calculated **before we receive** any signal

a ← **b**

Introducing the Tree of Promise



When receiving data

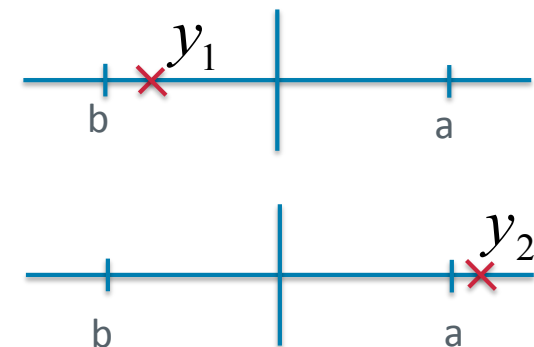
1st most promising path 2nd most promising path

$$x_1 = b$$

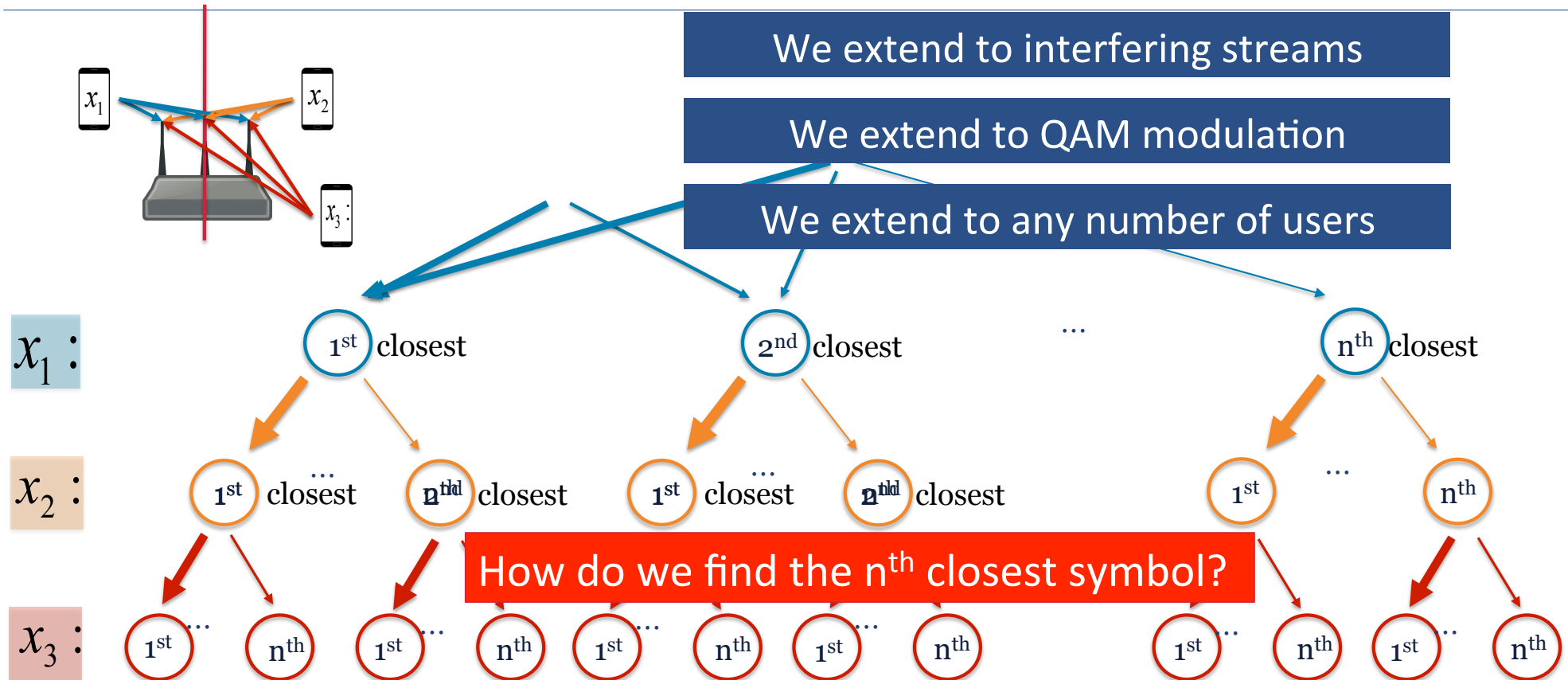
$$x_1 = a$$

$$x_2 = a$$

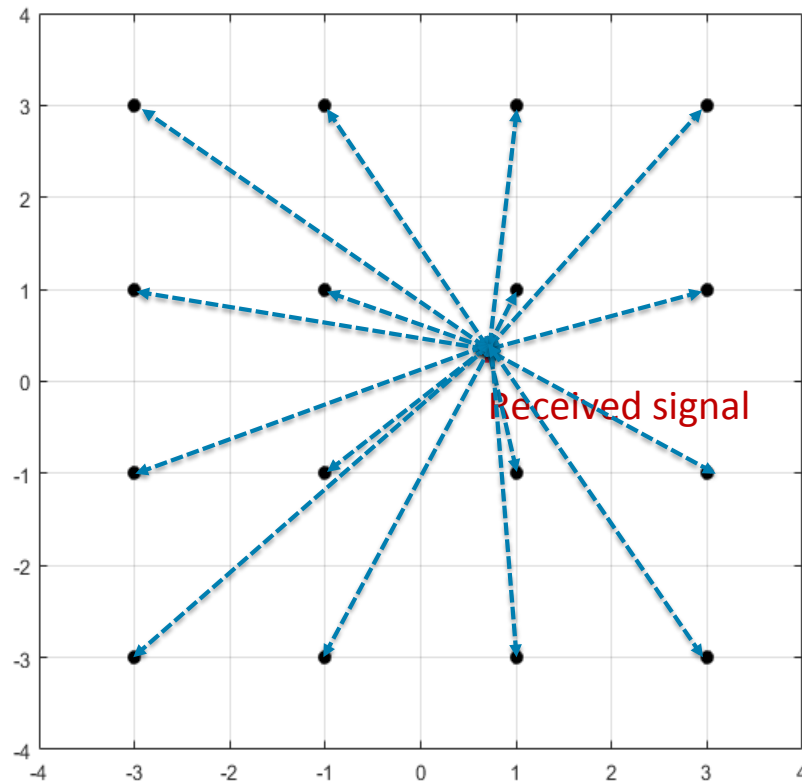
$$x_2 = a$$



FlexCore's Tree of Promise



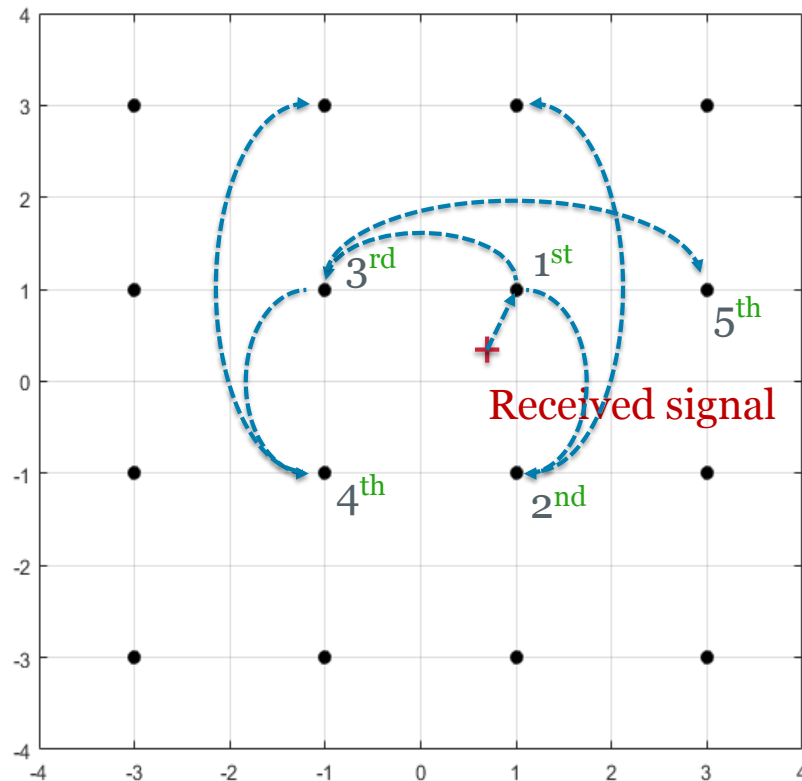
New Challenge: How do we find the n^{th} closest symbol ?



- Exhaustive Distance Calculation

Do we have better solutions ?

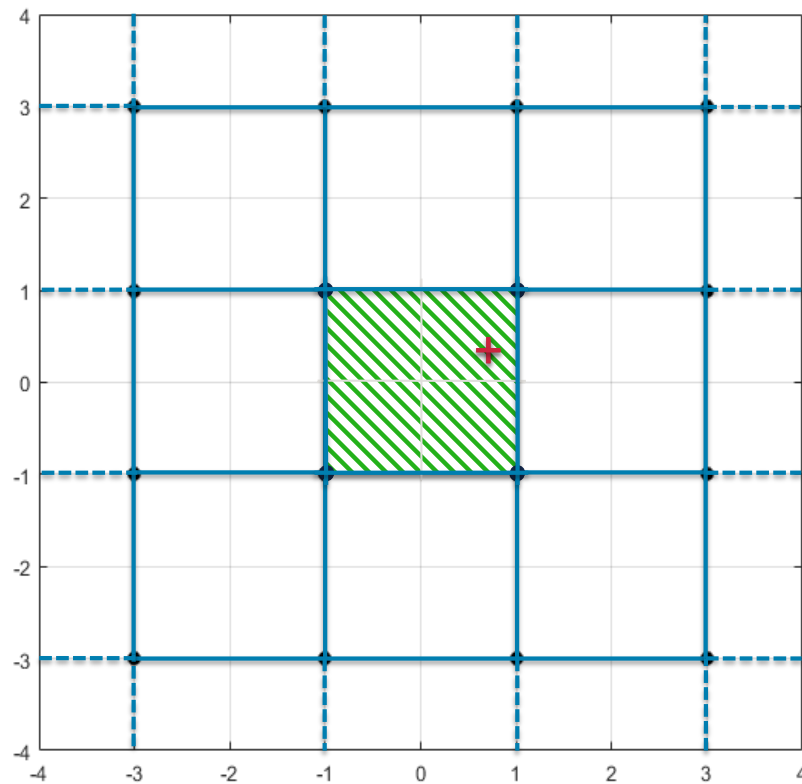
New Challenge: How do we find the n^{th} closest symbol ?



- **2-D ZigZag** [Geosphere, Sigcomm '14]

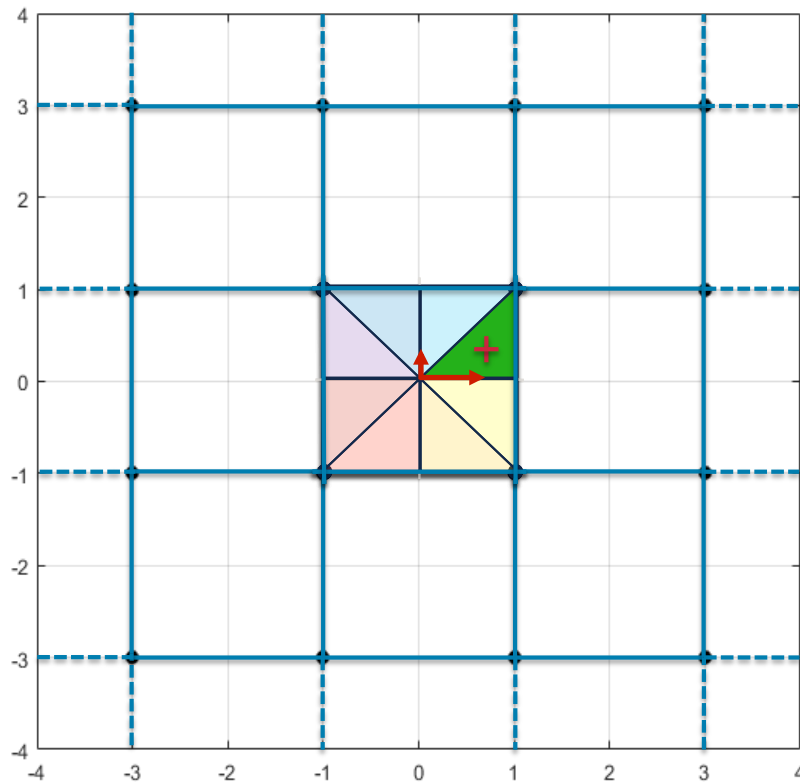
Can we find it without exhaustive or sequential search?

FlexCore's symbol selection



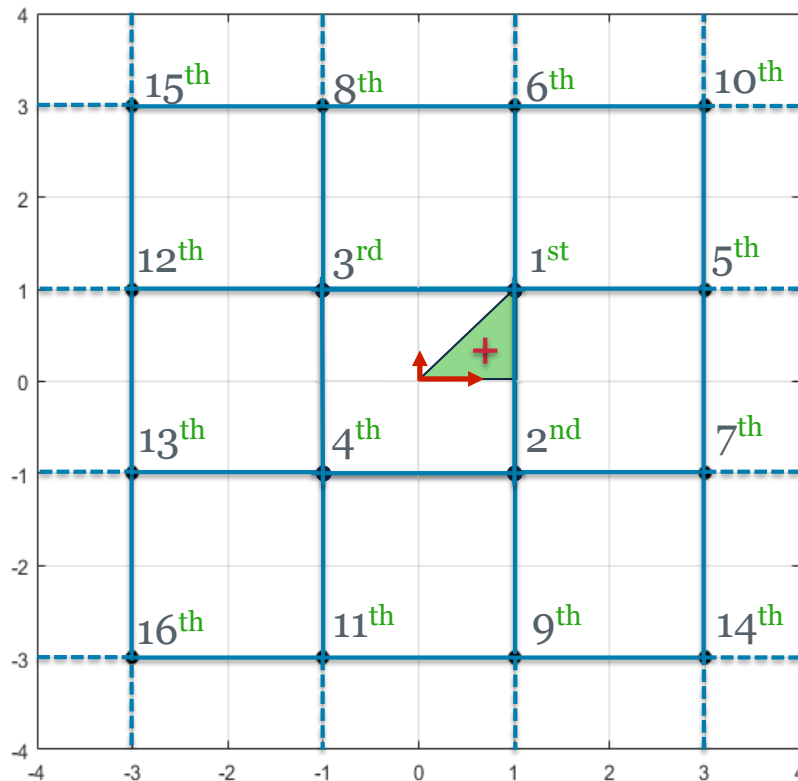
1. Find via \leq **comparisons** the square of the received signal.

FlexCore's symbol selection



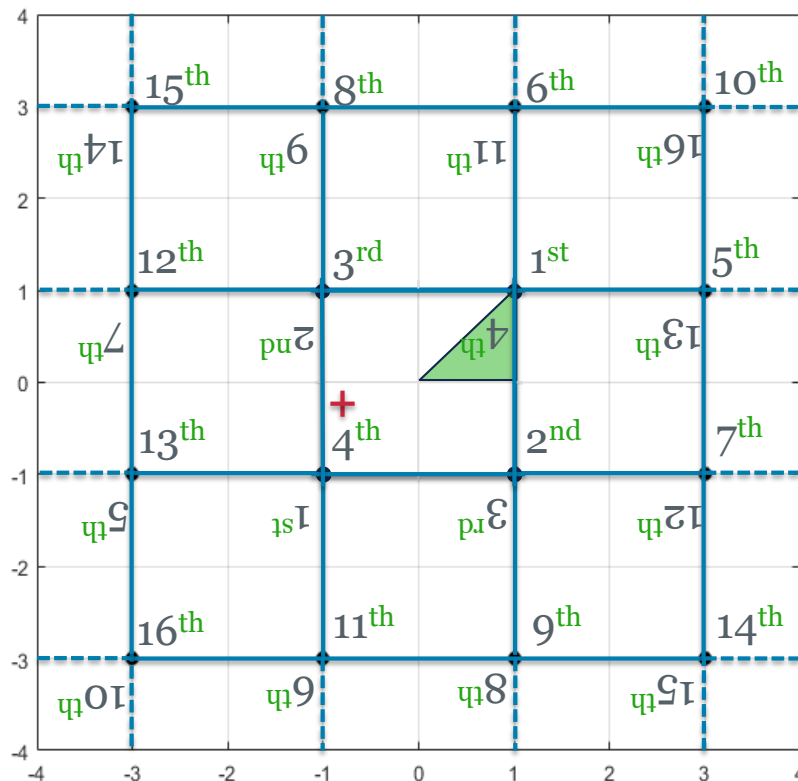
1. Find via \leq **comparisons** the square of the received signal.
2. Refine received signal to one of eight possible triangles, via \leq **comparisons**.

FlexCore's symbol selection



1. Find via \leq **comparisons** the square of the received signal.
2. Refine received signal to one of eight possible triangles, via \leq **comparisons**.
3. Use approximate pre-defined order for all the triangles, that is
 - stored in a **look-up table**

FlexCore's symbol selection



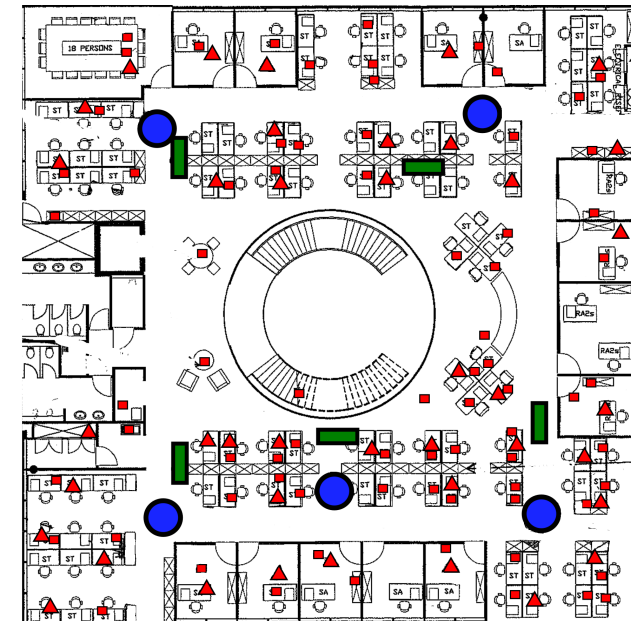
1. Find via \leq **comparisons** the square of the received signal.
2. Refine received signal to one of eight possible triangles, via \leq **comparisons**.
3. Use approximate pre-defined order for all the triangles, that is
 - stored in a **look-up table**
 - same for all triangles due to the QAM **symmetry**

Evaluation Questions

1. Is FlexCore's data throughput elastic with the number of PEs?
2. Does FlexCore offer better throughput and complexity compared to state-of-the-art approaches [FCSD, TCOM '08] of similar latency?
3. Is FlexCore capable of supporting real-time LTE detection on commodity GPUs?
4. Can FlexCore realize energy efficiency gains for a given throughput target?

FlexCore's Implementations and Evaluation Methodology

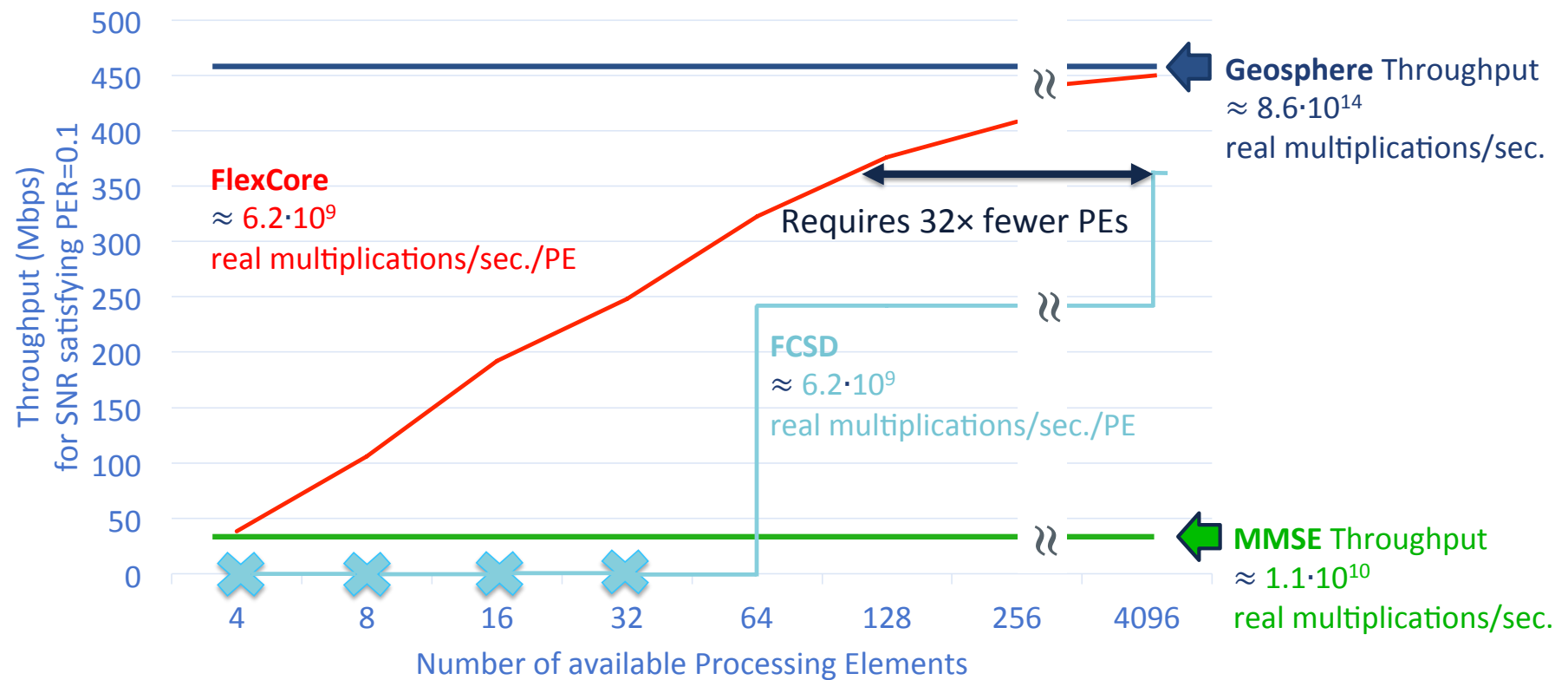
- Implemented FlexCore on **FPGA** [Verilog], **GPU** [CUDA/C], **CPU** [OpenMP/C].
- Evaluate using **over-the-air experiments** and **channel trace-driven simulations** [Rice, WARPv3].
- Experiments are conducted in **an indoor environment** on the 5 GHz ISM band over a 20 MHz bandwidth.



In blue: 8-antenna access points
In green: 12-antenna access points
In red: single-antenna users

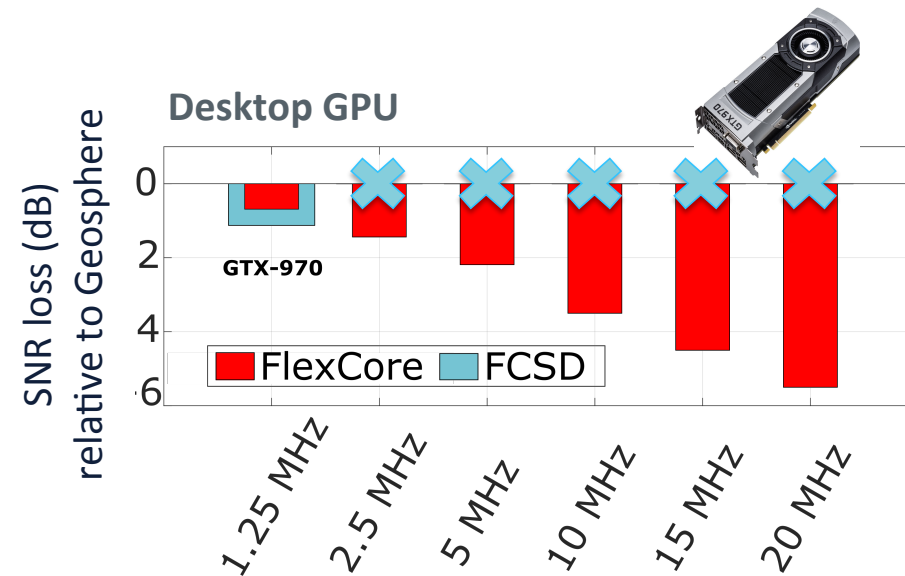
Elasticity of Throughput

12x12 MIMO, 64-QAM, 20MHz and $PER_{ML}=0.1$ (SNR=20.5 dB)



Real Time GPU-based LTE detection

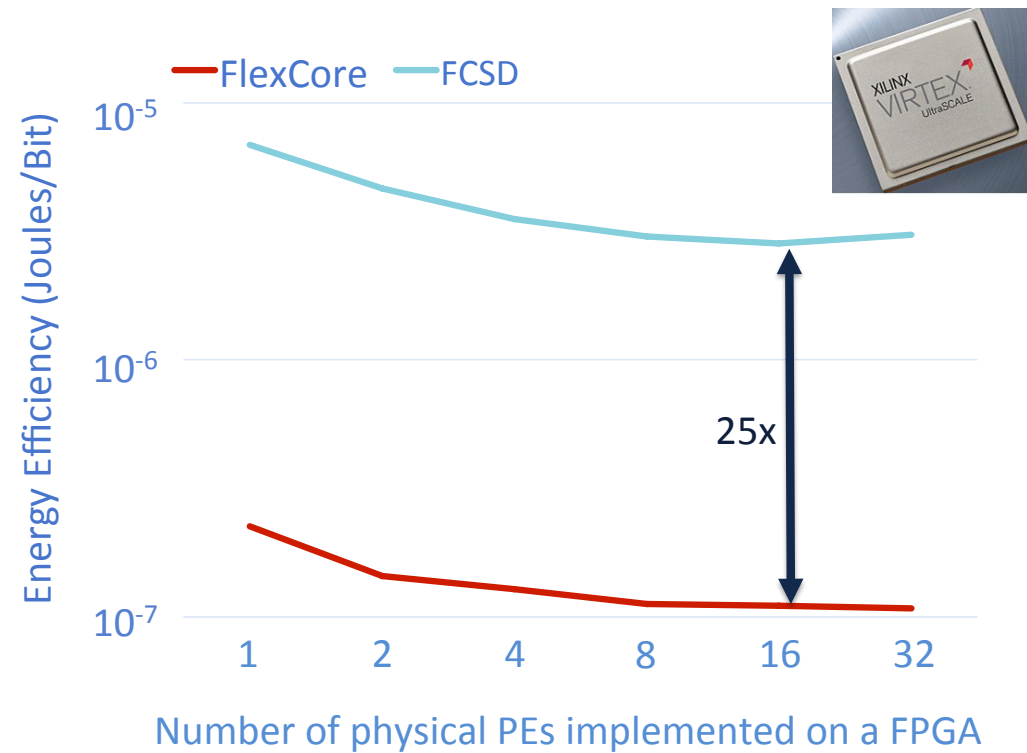
12x12 MIMO, 64-QAM, 20MHz and $PER_{ML}=0.1$ (SNR=20.5 dB)



FlexCore supports real-time MIMO detection for **12x12** LTE systems, using **commodity GPUs**.

FlexCore's Energy Efficiency

12x12 MIMO, 64-QAM, 20MHz and $PER_{ML}=0.1$ (SNR=20.5 dB)



Related Work

	Ability to focus processing	Parallelism granularity	Interaction between parallel components
Depth-first sphere decoders [Geosphere, SIGCOMM '14] [Burg, JSSC '05]	none	limited	Very high
Breath-first sphere decoders [Chen, VLSI '07], [Guo, JSAC '05]	none	limited	Very high
Fixed Complexity (FCSD) [FCSD, TCOM '08]	none	limited	Low
FlexCore	strong	high	Low

Related systems: Argos [Shepard, MobiCom '12], BigStation [Yang, SIGCOMM '13] currently employ linear detection schemes

Conclusion

- FlexCore **massively parallelizes** the **fundamental** maximum likelihood detection problem.
- **Focuses** processing to scale network throughput, efficiently utilizing **any number** of processing elements.
- FlexCore realizes **substantial power reduction** compared to state-of-the-art.

Funding Acknowledgments:

