Car-to-car Communication, Cognitive Radio and related Parallel Processing

15. July 2008

Edmund Coersmeier

NOKIA

Overview

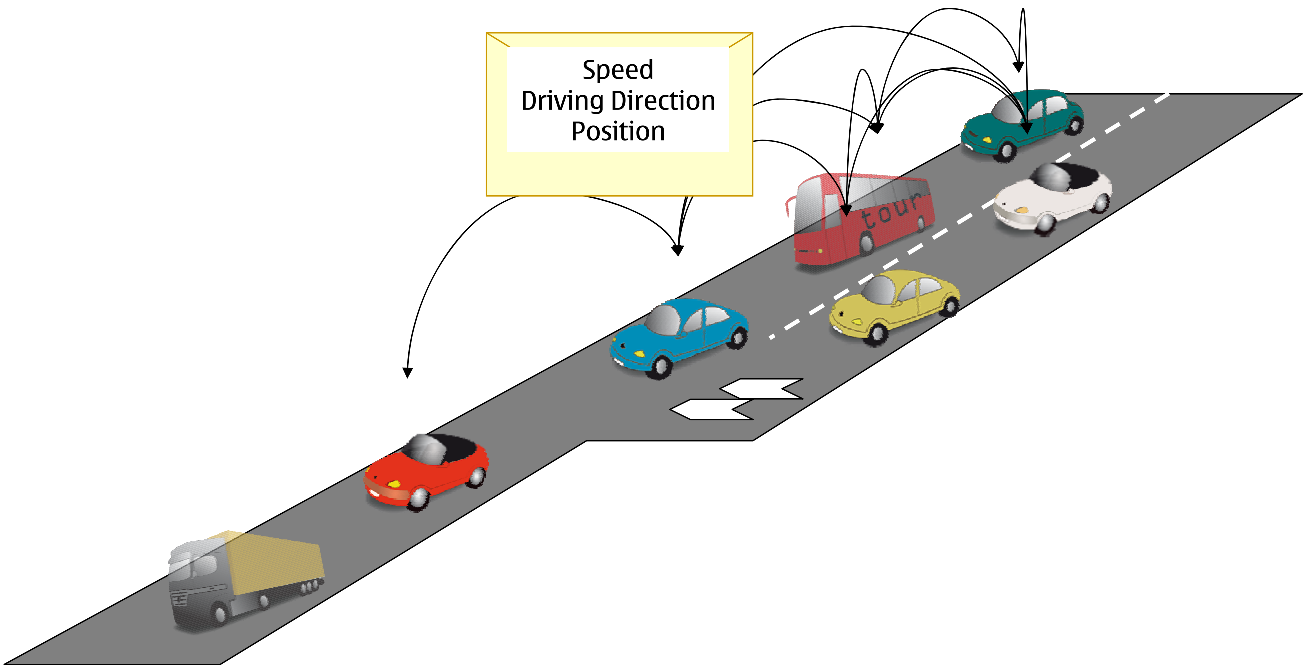
- Car-to-car communication
- Cognitive Radio
- Software Radio
  - Channel estimation processing
  - Multicore processing
- Conclusion

NOKIA

Car-to-car communication

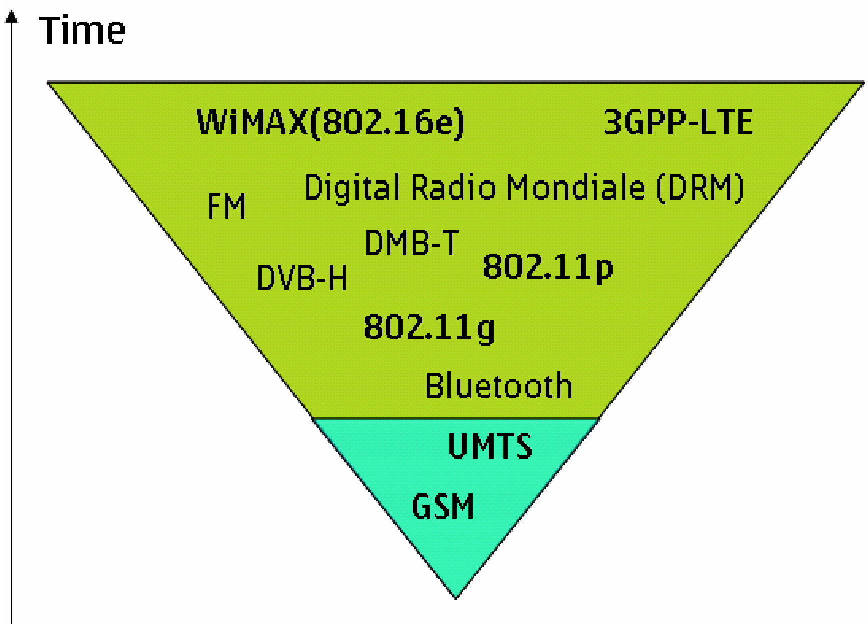
Basic idea

- Automatic car-to-car communication

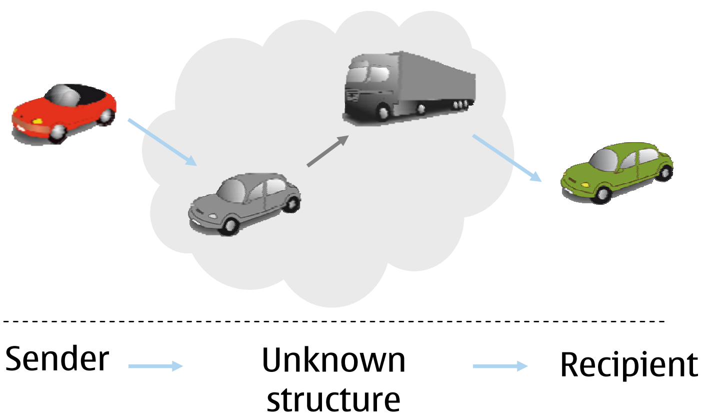


Related Technologies

- Starting with established wireless technologies
- Include step-by-step latest wireless systems

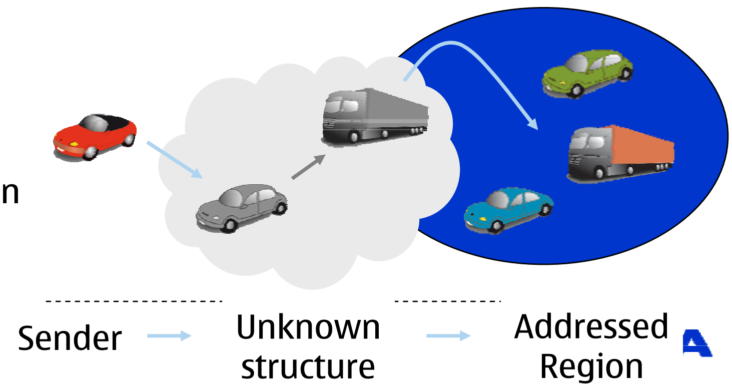


Unicast versus Geocast



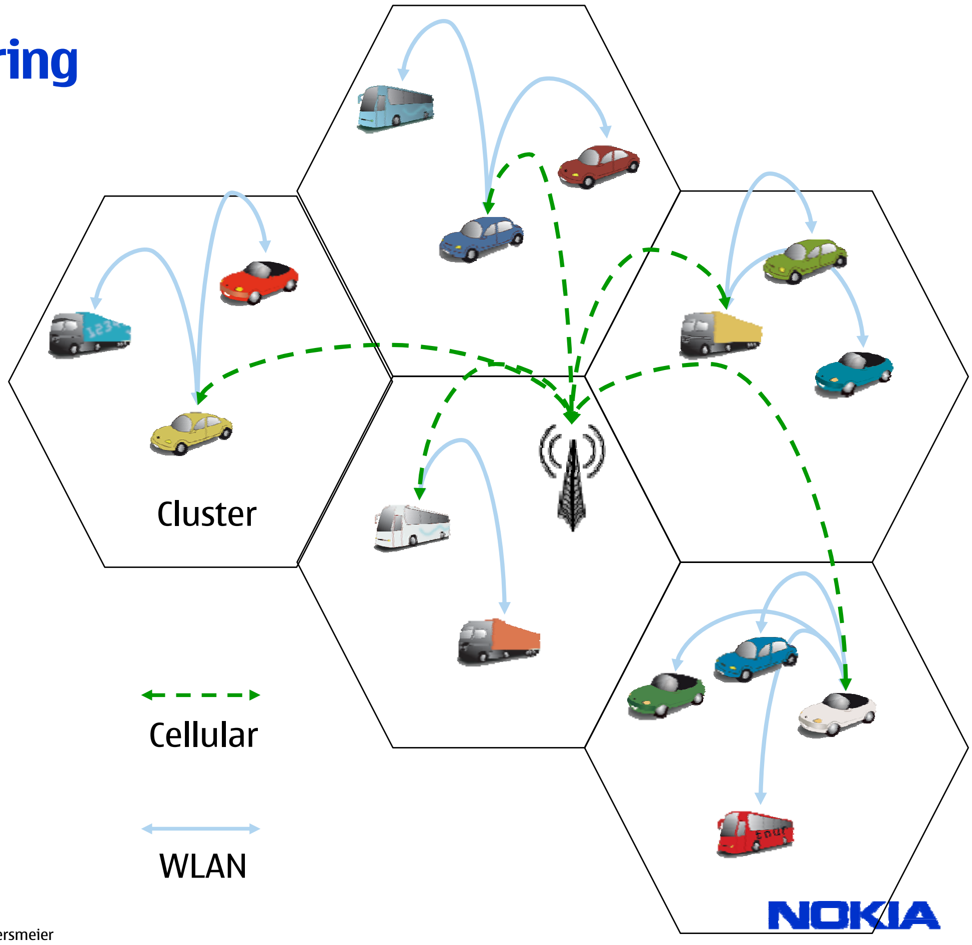
- Unicast
  - communication between two parties

- Multicast
  - several parties receive the same information
  - Geocast for car-to-car communication



Geographic Clustering

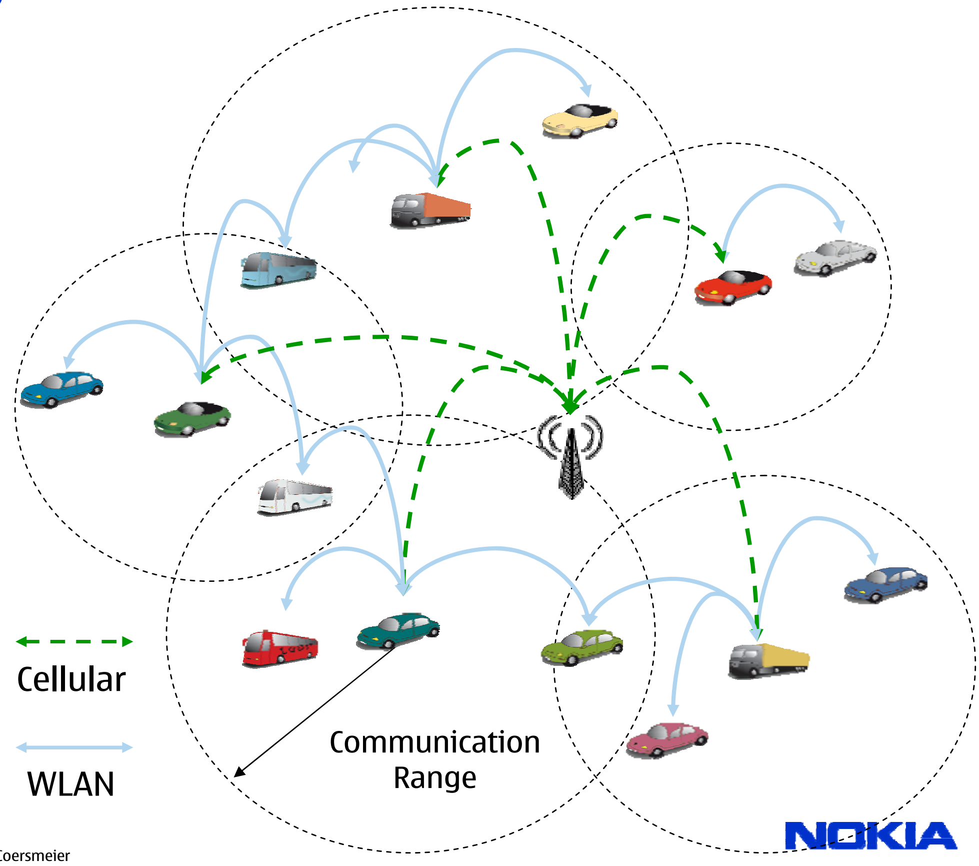
- Clusters mapped on landscape
- GPS to link parties to the corresponding clusters
- One car acts as cellular uplink node
- Other cars communication via WLAN



NOKIA

Dynamic Gateway

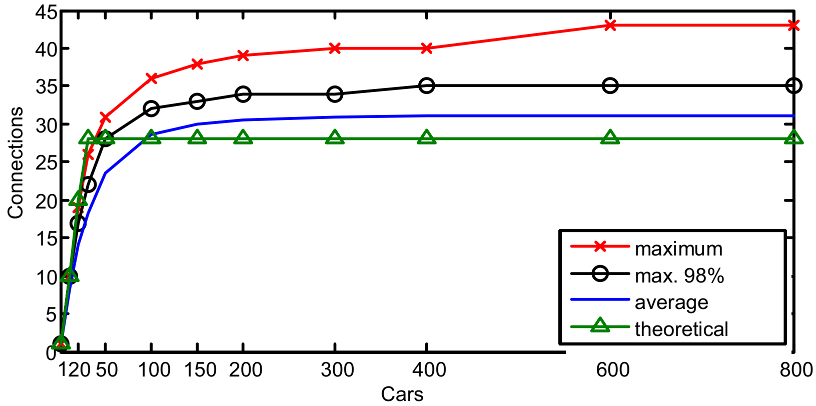
- No GPS positioning
- Cellular uplink node is established, when no other uplink is available
- Advantage
  - Communication range improved



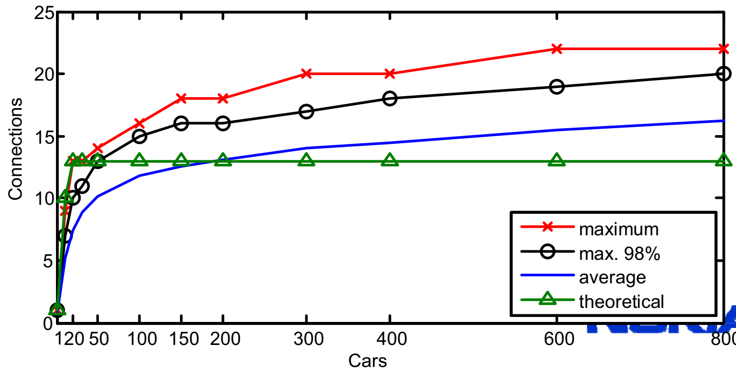
NOKIA

Required cellular connections

- Both setups converge with increasing number of cars



- Geographic clustering
  - Required UMTS connections
  - Highway scenario
  - Max = 43 connections



- Dynamic gateway
  - Required UMTS connections
  - Highway scenario
  - Max = 22 connections

Conclusion on Car-to-Car communication

- WLAN and latest cellular technologies can initiate car-to-car communication for consumer applications
- Dynamic Gateway approach more efficient than Geocast approach
- Both approaches are stable when number of cars increase

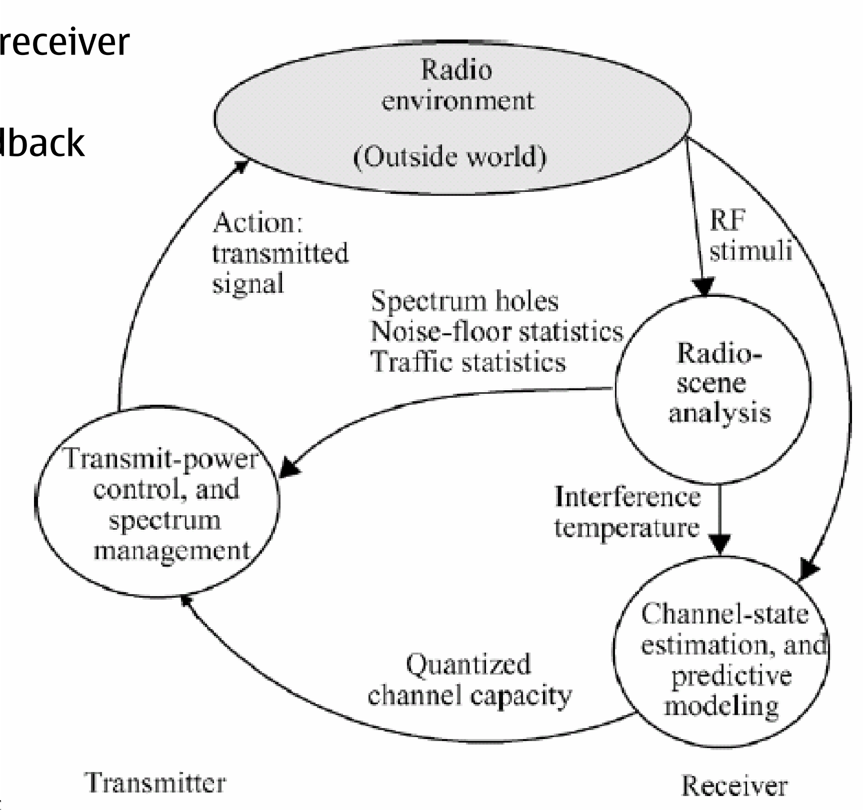
Cognitive Radio

Definition

- Cognitive Radio approach intends to efficiently utilize the limited wireless spectrum
- Cognitive Radio scans the spectrum and decides which spectrum is the best applicable for the corresponding wireless technology
- Cognitive Radio approach is based on three cognitive tasks
  - Radio-scene analysis (receiver)
  - Channel-state estimation and predictive modeling (receiver)
  - Transmit-power control and spectrum management (transmitter)

Cognitive Cycle [source: Simon Haykin]

- Transmitter and receiver interact
- -> Real-time feedback channel



Motivation

Requirement when combining Cognitive and Software Radio

- Cognitive Radio for spectrum efficiency
  - analyzing user application
  - definition of wireless requirements
  - spectrum scanning
  - definition of radio characteristics

- Software Radio
  - adjusts transmitter and receiver algorithms
  - transforms algorithms to an applicable architecture
  - maps the architecture on available processor platform
  - balances between different, parallel operating radios

- To achieve efficient receiver implementations Software Radio requires
  - strong flexibility in terms of
    - algorithm complexity
    - power consumption
  - *support from Cognitive Radio*

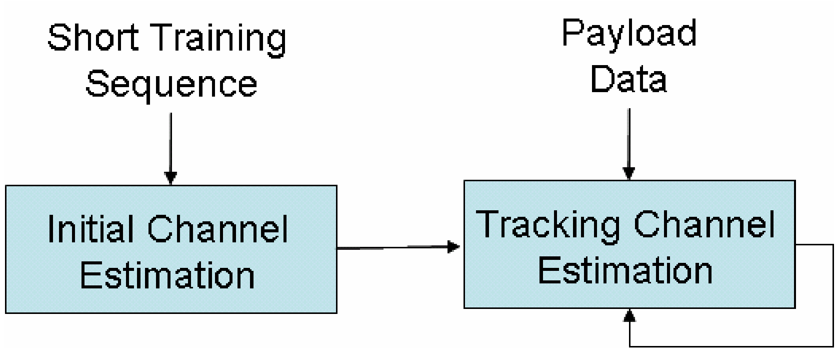
Cognitive Radio Enhancement

Channel-State Estimation

- Channel-State Estimate to judge about channel capacity

- Semi-blind training
  - Supervised training mode via short training sequence
  - Tracking via data feedback

- Rate feedback to transmitter to setup
  - data rate
  - transmit-power control

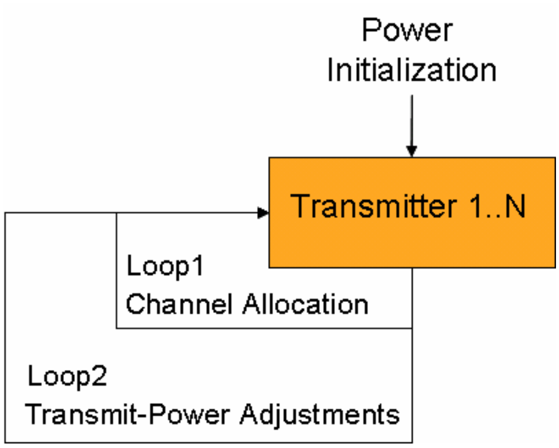


Transmit Power Control

- Power initialization

- Inner Loop
  - Allocation of a number of channels

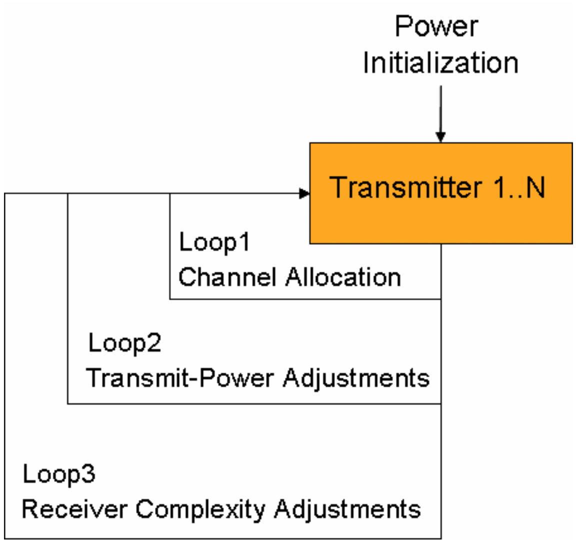
- Outer Loop investigates achieved data rate
  - exceeding
  - matching
  - undershooting



- Outer Loop adjusts the transmit power of each transmitter
  - All transmitters run from data-rate perspective with optimal transmit power
  - *What is about the receiver complexity?*

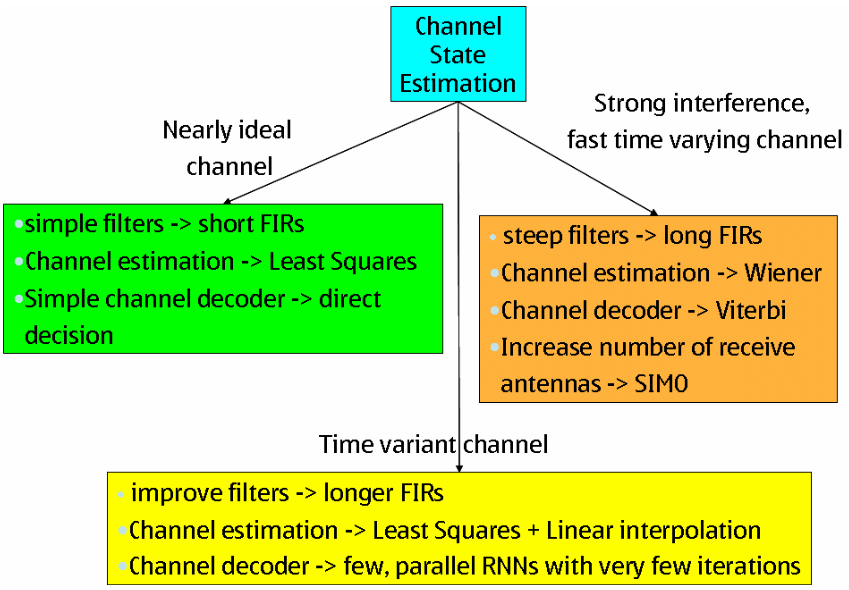
Cognitive Radio Enhancement

- Each receiver includes an option to ask for low receiver complexity
  - Transmit-power increase
  - High quality channel selection
- Transmit-power increase
  - Other transmitters reduce power
  - Other receivers increase complexity
- High quality channel selection
  - Find a better fitting free channel
  - Exchange already allocated channels



Receiver Algorithms with different Complexities

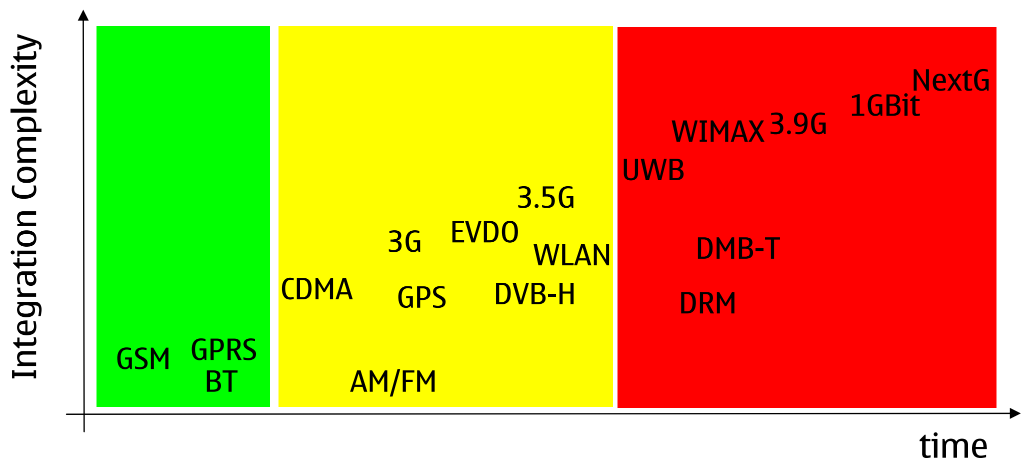
- Different receiver complexities based on channel-state estimation
- Receiver complexities can change at any time



Multicore Processing for Software Radio Architecture

Motivation

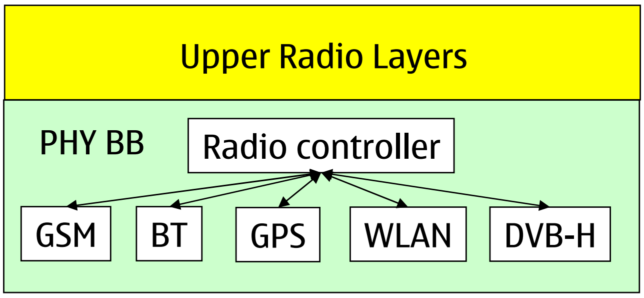
- Number of wireless technologies rises quickly



- Phone form factors limit the number of dedicated chip sets
- Different mobile device use cases require different technology combinations
- Implementation costs are high if dedicated digital baseband chips are used

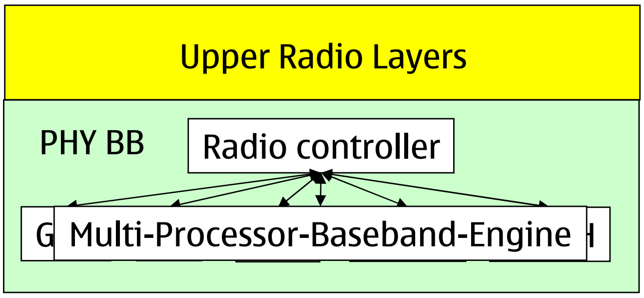
How to reduce number of dedicated digital baseband chips?

- Starting position



- Idea – replace digital dedicated baseband chip sets by multi-processor platform

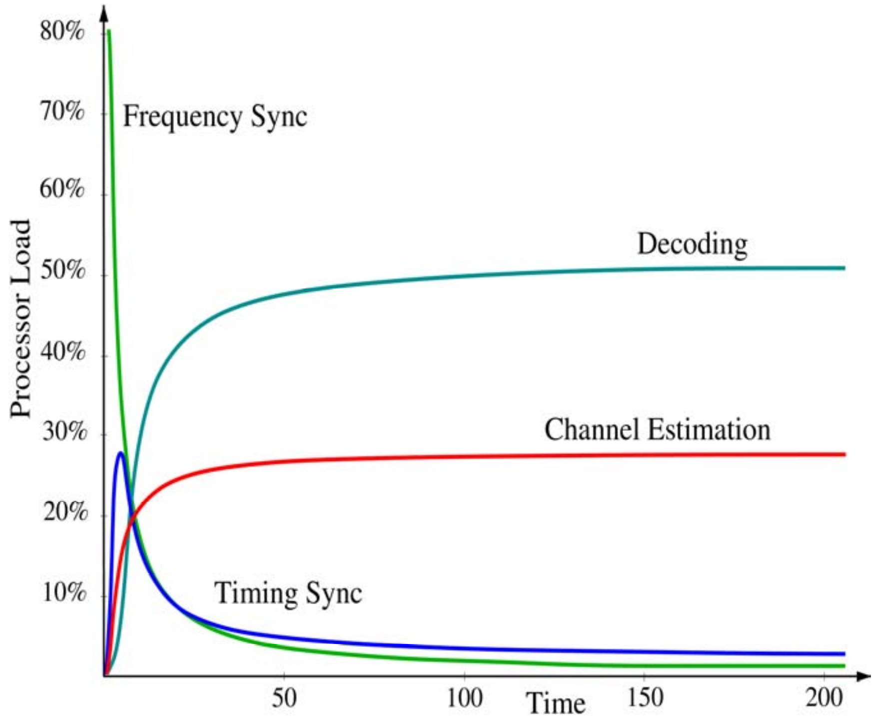
- Solution



Channel Estimation Processing

Processor Load as Measure for Algorithm Complexity

- OFDM PHY layer algorithms running on a single DSP (here Digital Radio Mondiale)
- Decoding and channel estimation require most processor load
- Future software radio implementations require more scalability of complex algorithms than today available
- Research question: how to scale radio algorithms from processor load perspective?



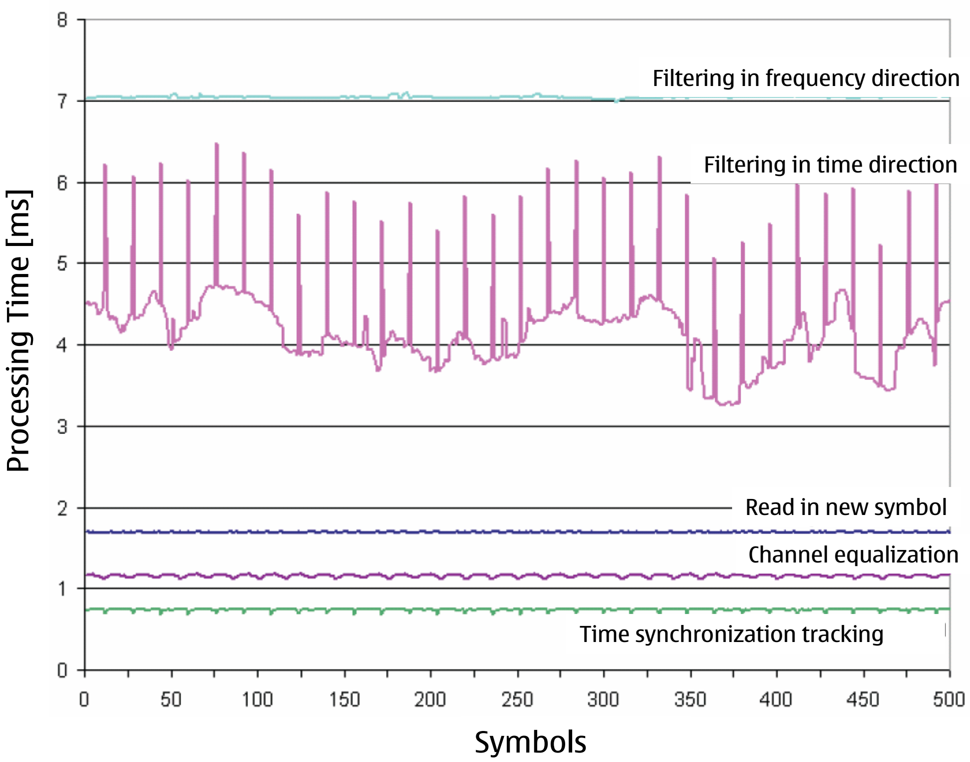
Channel Estimation via Wiener filter

- The channel transfer function $\hat{\mathbf{H}}$ can be interpolated from pilot carriers using the Wiener filter equation

$$\hat{\mathbf{H}} = \mathbf{R}_{HH_p}\left(\mathbf{R}_{H_pH_p} + \frac{\beta}{SNR}\mathbf{I}\right)^{-1}\mathbf{H}_p$$

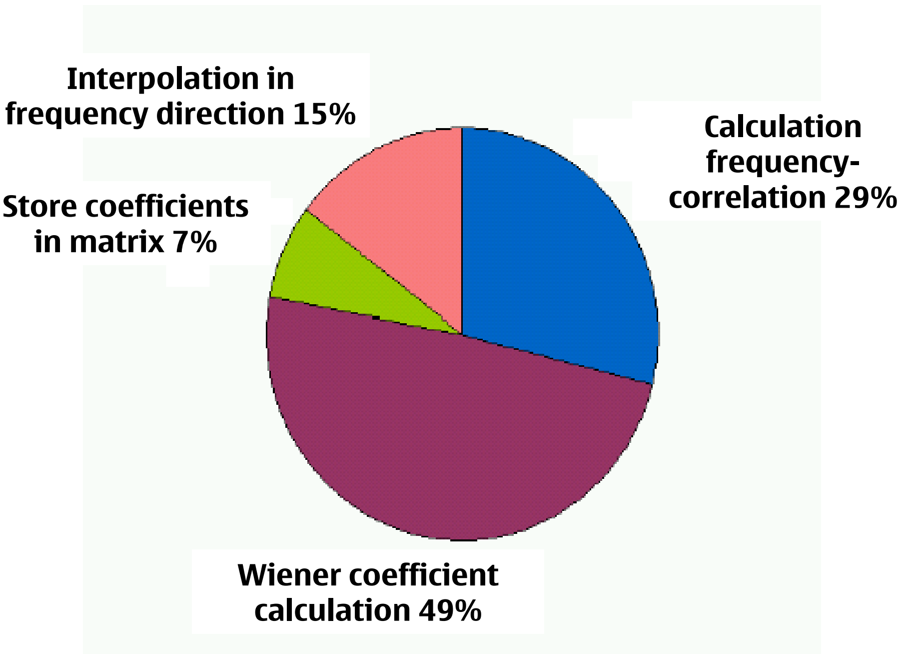
- Performance can be improved if filter coefficients are computed online (no pre-computation)
- In this case matrix inversion is the most time consuming task

Processing for Channel Estimation



Processing in Frequency Direction

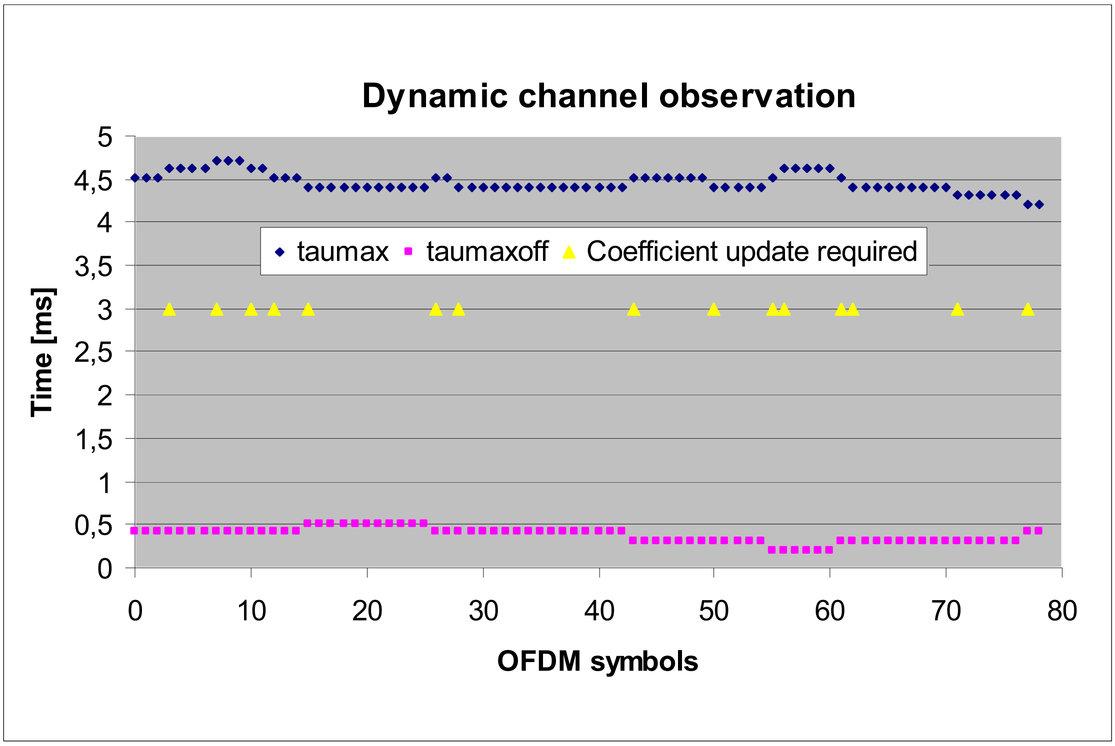
- Coeff. update requires most processor load.
- Only 15% for interpolation.



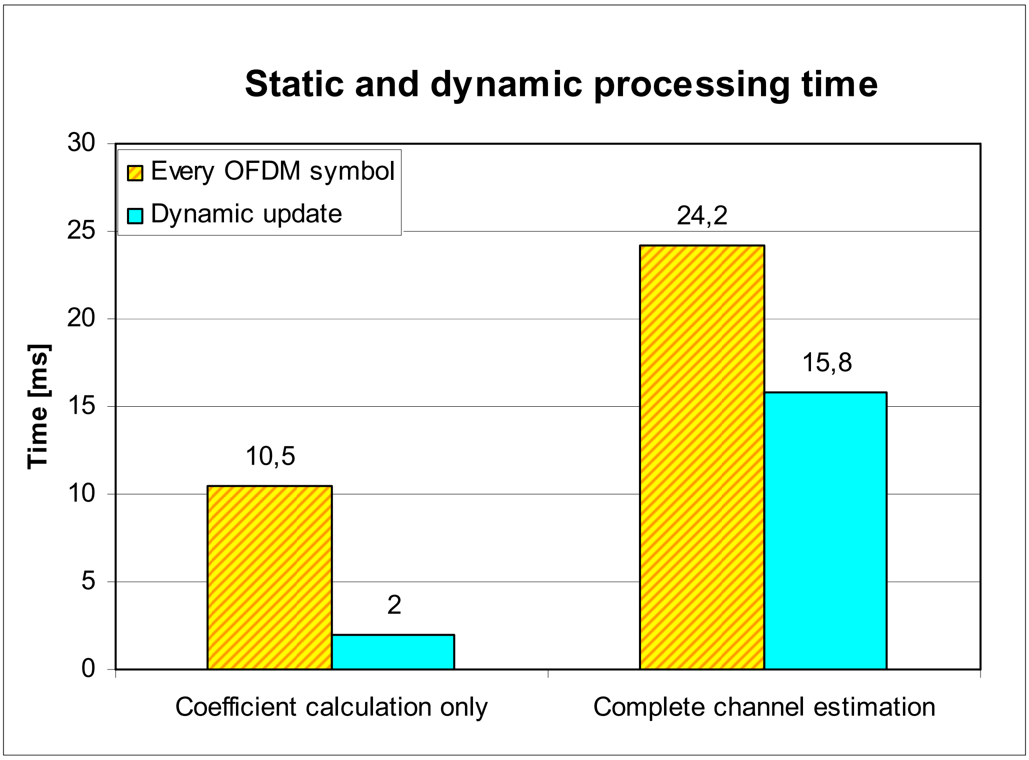
Channel Estimation Optimization

Changing Channel Properties

- τ_{max} = max. excess delay.
- τ_{maxoff} = impulse offset.



Reducing Processor Load



Conclusion – Channel Estimation

- Channel estimation based on run-time coefficient calculation is complex
- Channel properties need to be monitored
- Dynamic coefficient update leads to significant SW radio power reduction

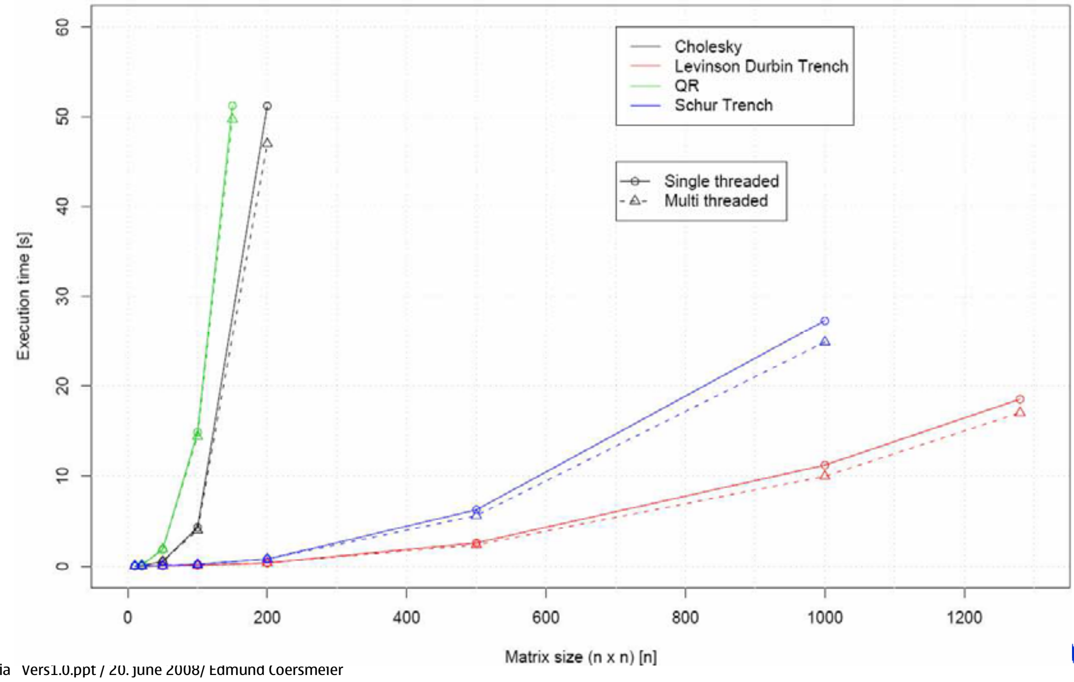
Mapping traditional Algorithms on Multi-Processor Platforms

Matrix Inversion

- Toeplitz matrix inversion is a key mathematic operation for online Wiener filter coefficient calculation to enable high performance channel estimation
- Evaluation is done in ARM Realview EB Rev. B MPCore platform
  - CT11MPCore
  - 4 x ARM11 CPU with 200MHz core frequency
  - L1 cache with 32kB memory for data, 32kB memory for instructions
  - unified L2 cache 1MB running at core frequency 200MHz (shared memory)
  - L220 cache controller
  - external bus frequency 20MHz
  - no floating point co-processors used
- Matrix inversions are based on C++ code with 64-Bit floating point arithmetic
- Integer modeling / word-length limits will *decrease relative speed-up* because
  - times between synchronization points will decrease
  - relative increase of synchronization overhead

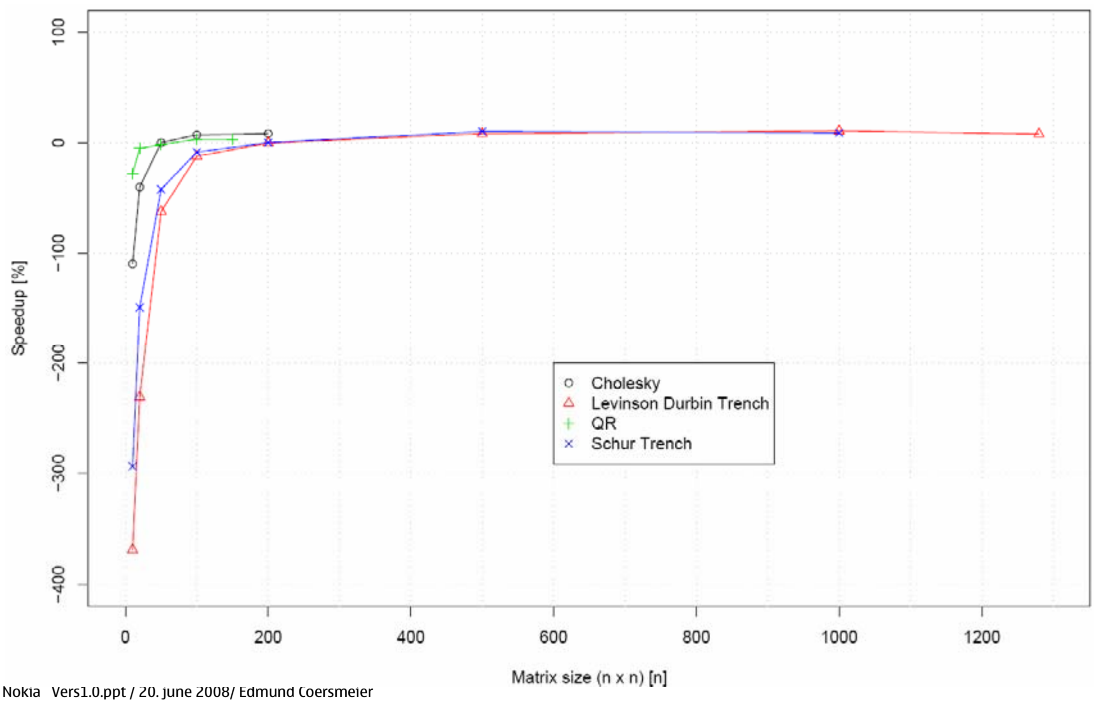
Single-CPU versus Multi-Cores

- ARM processors scale execution time to the expected operation count
- Cholesky, LDT, QR are separated into 4 cores, ST includes only 2 cores
- Only inversion of large matrices (> 100x100) is improved a little bit from speed perspective



Speed up through Multi-Threading

- Speed up from -360% up to only +10.7% whereas 400% theoretically possible
- Algorithms can slow down when matrix calculation is faster than synchronization
- Most complex algorithms QR and Cholesky most significantly speed up



Analyzing Matrix Inversion Results

- The longer math calculation versus thread synchronization time the better speed up

- For large matrices, calculation starts to be longer than synchronization time
  - But speed up is far from factor 2 or 4 which could be theoretically possible
  - Data interdependency inside the algorithms are high

To utilize multi-processor platform processing power

a significant change of data dependencies is required

NOKIA

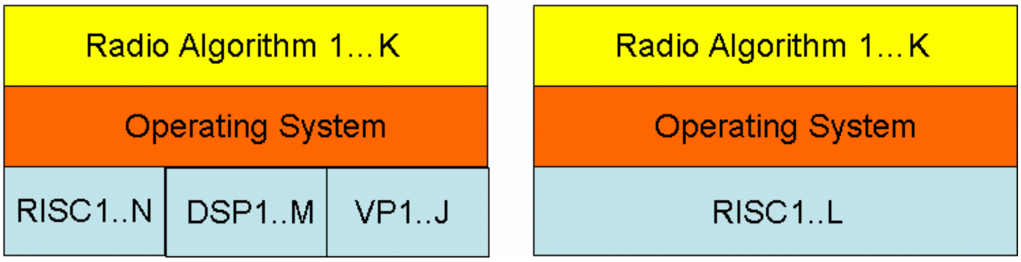
Introducing scalability to software radio algorithms

NOKIA

Receiver scaling through multi-processor platforms

- Pure hardware-optimized design can be replaced by multi-processor platform
- Several radios and their algorithms run in parallel
- The same radio functions need to optimally fit on different processor types

A change of mathematics fitting to different processor types is required



Example of parallel radio algorithms – channel decoding

- Viterbi
  - high signal processing performance
  - optimal for hardware implementation
  - sub-optimal for software radio approach
  - difficult to parallelize
- Recurrent Neural Networks
  - do not outperform Viterbi signal processing performance
  - similar mathematics as adaptive filters
  - easy to parallelize several networks

$$\min_{\mathbf{w}(n)} \|e(n)\|_2^2 = \min_{\mathbf{w}(n)} \|\mathbf{X}(n)\mathbf{w}(n) - \tilde{y}(n)\|_2^2$$

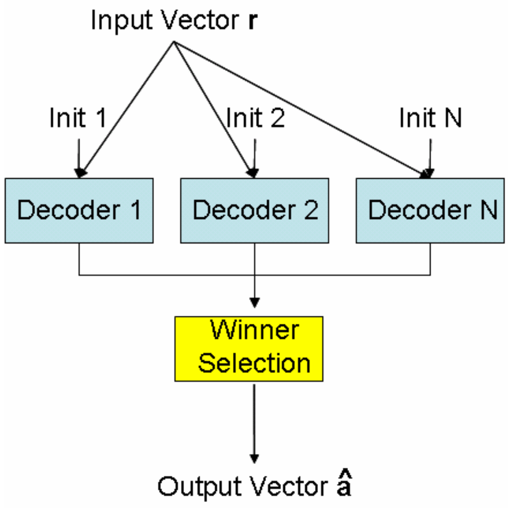
Least-Squares - Adaptive Filter

$$\min_{\mathbf{c}} \|\mathbf{e}\|_2^2 = \min_{\mathbf{c}} \|\mathbf{r} - \mathbf{c}\|_2^2 = \min_{\mathbf{a}} \|\mathbf{r} - \mathbf{G}^T \mathbf{a}\|_2^2$$

Recurrent Neural Network

N-parallel channel decoders

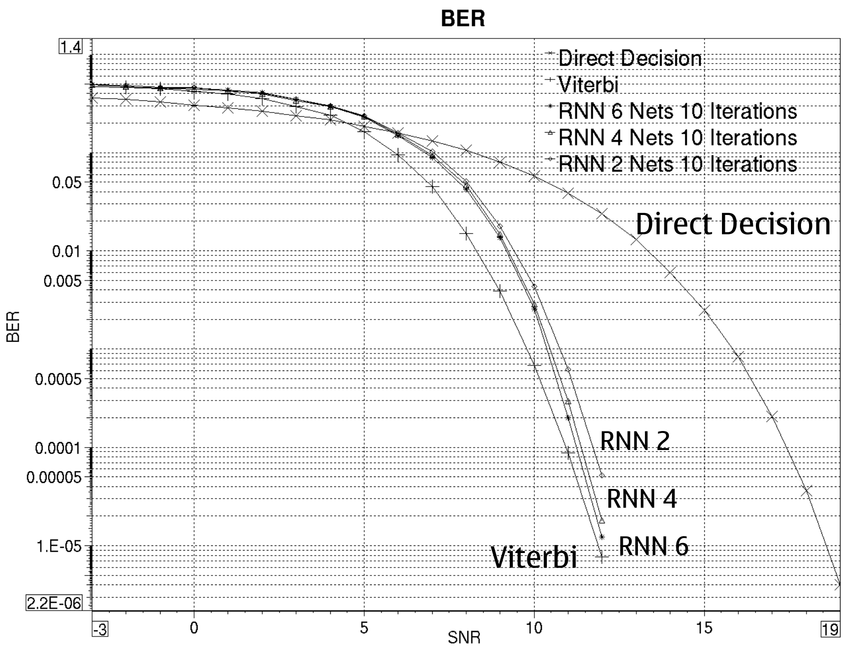
- Run several networks in parallel
- The more networks, the higher the channel decoding performance
- Each network
  - operates independently of all others
  - works on the same data set
- Research topic – optimize complexity of each channel decoder network



Scalability introduces flexibility for the multi-processor platform processor load

Simulation results for recurrent neural networks

- Number of Recurrent Neural Networks can be adjusted to channel quality
- Optimal design approach for multi-processor platforms



Acknowledgment

The presented work was partly carried out within the German funded BMBF-project

3GET - 3G Evolving Technologies
No. 01 – BU - 356

The authors have the responsibility for the content of this presentation

Conclusion – Multicore Processing

- Parallel processor platform should be able to replace optimized hardware
- *To utilize multi-processor platform processing power a significant change in algorithm data dependencies is required*
- Software Radio needs to handle several radios in parallel
- *A change of mathematics fitting to different processor types is required*
- *Radio functions need to be scalable* to balance
  - required system performance
  - multi-processor platform load