Wireless Networks In-the-Loop: Software Radio as the Enabler

Jens Elsner, Martin Braun, Stefan Nagel, Kshama Nagaraj and Friedrich Jondral

Software Defined Radio Forum Technical Conference, Washington DC, December 3, 2009

Communications Engineering Lab
Prof. Dr.rer.nat. Friedrich K. Jondral

Overview

Wireless Networks In-the-Loop: Software Radio as the Enabler

- Introduction: Network Simulators and Software Radio
- Design Concepts
- Implementation
- Sample Network
- Conclusion and Future Work

Overview

Introduction

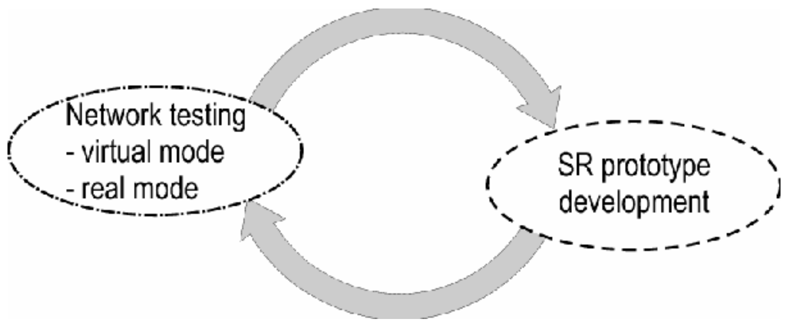
Software Radio techniques allow for loop-testing of heterogeneous networks.

Design of wireless networks	Software Radio Techniques

▪ Design requires theoretical analysis and computer-based Monte Carlo method simulations
▪ Simulation is done in parallel to design
▪ Dedicated tool: network simulators
	▪ Iterative model-based design stages, e.g.,
	  ▪ Platform independent model
	  ▪ Platform specific model
	  ▪ Executable

Wireless Network In-the-Loop

- Simple idea: Use Software Radio code / model in simulation
- Offer virtual mode / real mode
- Use production code in simulation, reproducible test environment and results



Overview

Wireless Networks In-the-Loop: Software Radio as the Enabler

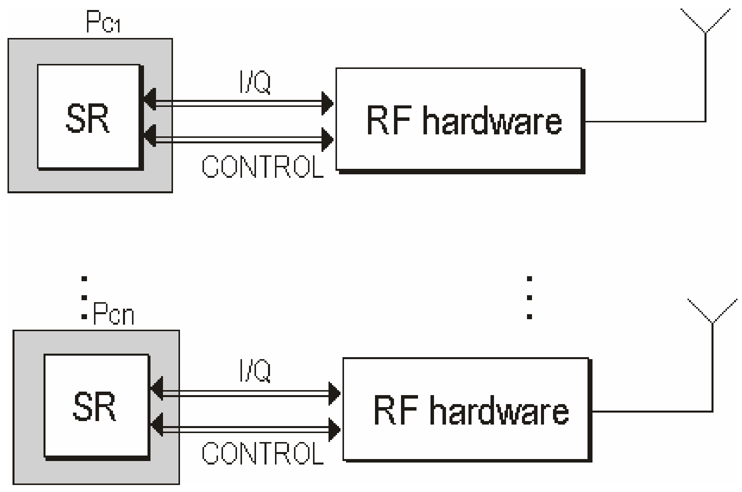
Design Concepts: Real Mode

Real mode needs standardized digital I/Q base band / RF software interface.

Real mode

- **Network in real environment:**
  - Different SR terminals running on different machines with different RF hardware
- **Components:**
  - SR terminal
  - RF hardware
  - Physical channel
- **RF hardware is addressed using a standardized software I/Q base band interface**
  - E.g. SDR Forum Transceiver Facility API or GNU Radio USRP API

System overview



See GNU Radio (http://gnuradio.org/trac) or E. Nicollet and L. Pucker, "Standardizing Transceiver APIs for Software Defined and Cognitive Radio," *RF Design*, Feb 2008, SDR Transceiver Facility Working Group

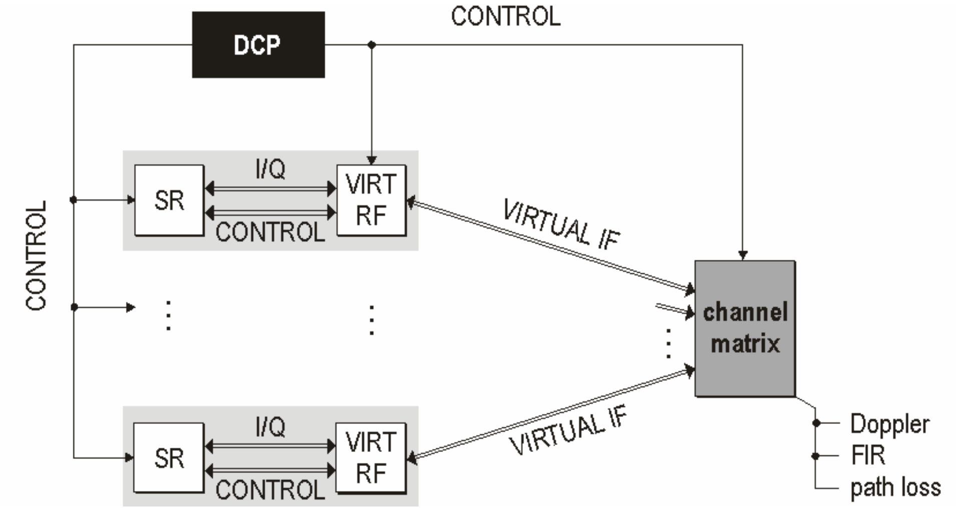
Design Concepts: Virtual Mode

Virtual mode additionally needs to abstract RF hardware and channel.

Virtual mode

- **Network in simulation:**
  - Components are run as separate processes, on same or separate machines
- **Components:**
  - **Dispatch / Control process**
  - **SR terminals**
  - **Virtual RF hardware**
  - **Channel matrix**
- **Virtual RF software API identical to RF software API**

System overview

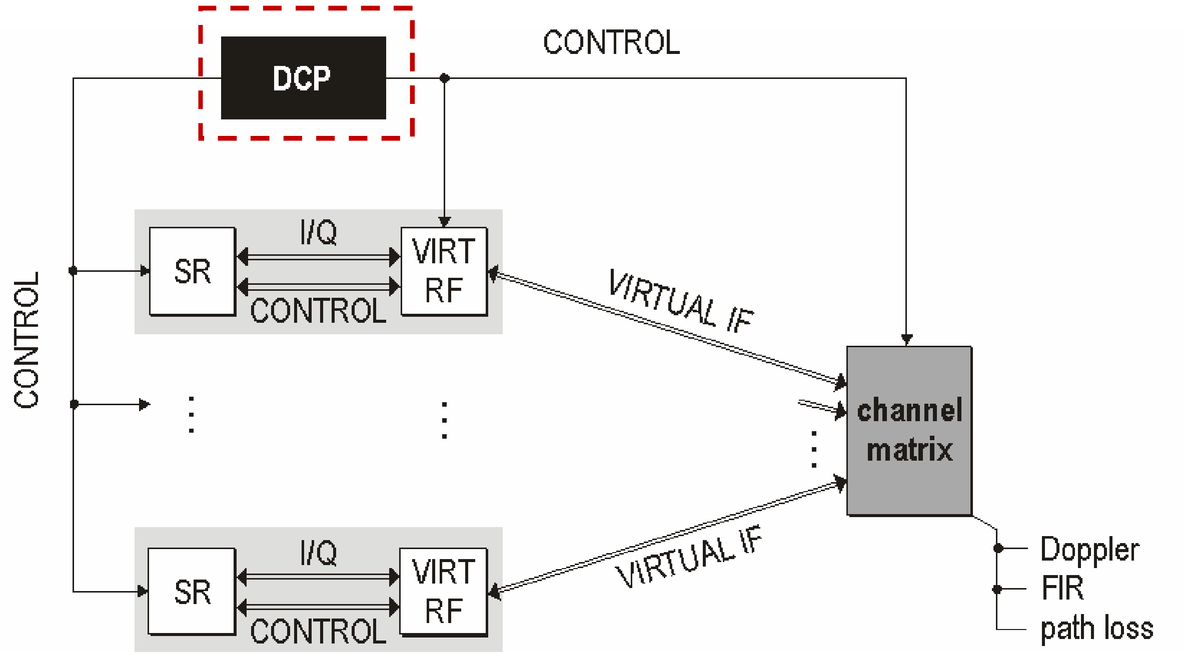


Design Concepts: Components Virtual Mode

DCP controls SR processes, provides world model and parameterizes channel matrix.

Dispatch and Control Process (DCP)

- **Initiates and controls the SR processes**
- **Responsible for selecting a channel model**
- **Constantly updates channel matrix process according to internal world model**

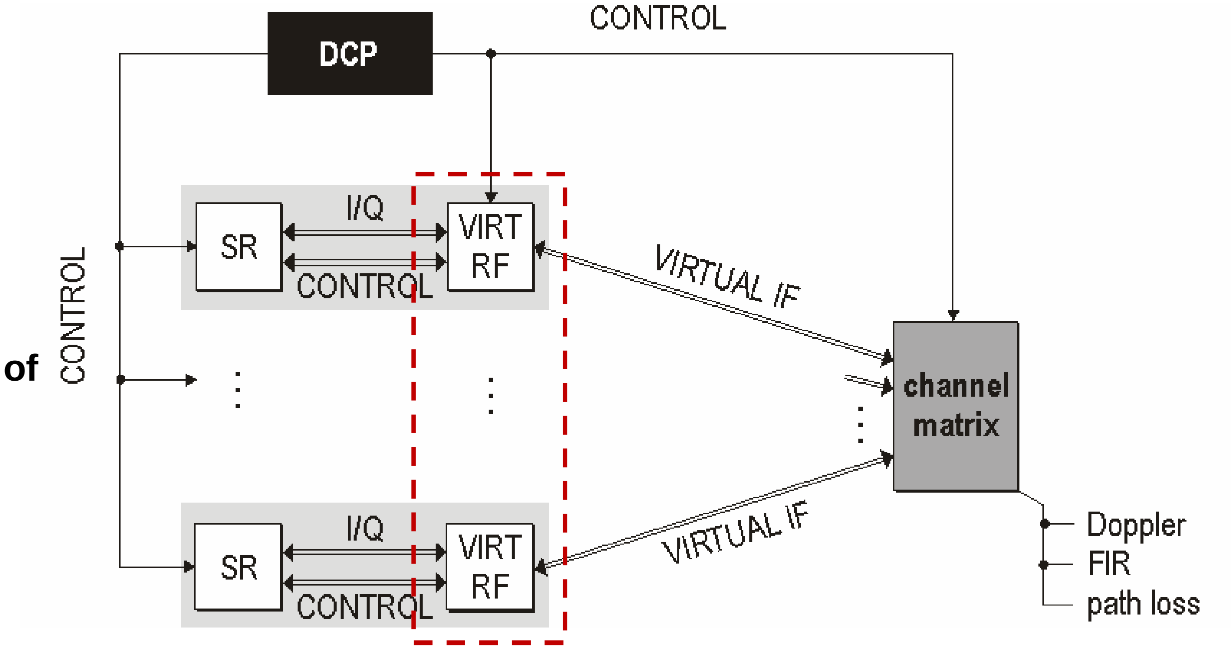


Design Concepts: Components Virtual Mode

Virtual radio front-end models analogue processing.

Virtual radio front-end (Virt RF)

- **Simulates signal path from I/Q base band to antenna**
- **Models non-idealities of analogue processing**

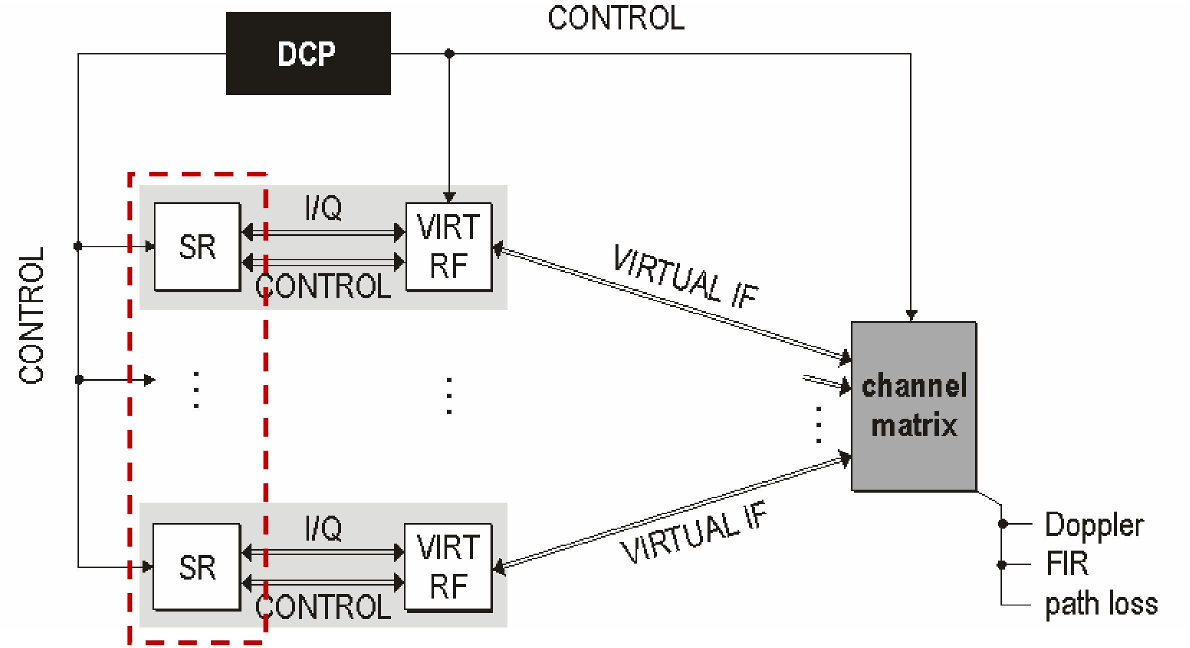


Design Concepts: Components Virtual Mode

SR processes provide I/Q base band processing, transmit and receive path.

Software Radio (SR) terminals

- **SR terminals process I/Q base band samples, send and receive**
- **Control (virtual) RF front-end parameters, such as frequency, gain, transmit power etc.**
- **SR terminal processes interact through virtual RF and channel matrix**

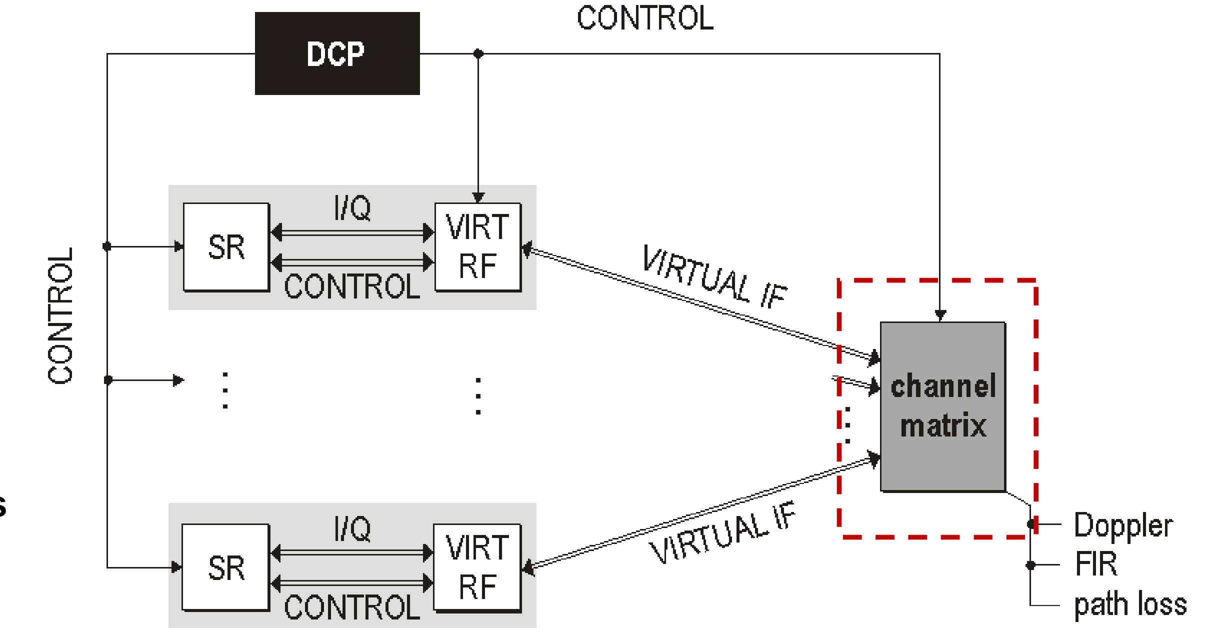


Design Concepts: Components Virtual Mode

The channel matrix models antenna to antenna wave propagation.

The channel

- **Channel matrix simulates the propagation path from antenna to antenna**
- **Parameterized by the DCP according to a world model**
- **Adds and redistributes I/Q streams from SRs**



Overview

Wireless Networks In-the-Loop: Software Radio as the Enabler

Implementation: Software / Hardware Tools

Choice of SR framework: GNU Radio and Ettus Research USRP.

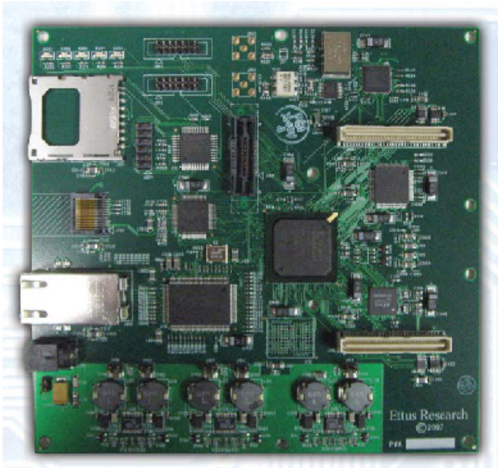
Tools

Virtual Mode Software

- SR terminals
- RF software API: USRP-based
- Channel

Real Mode Software and Hardware

- Ettus USRP1 or USRP2
  - GNU Radio Python USRP interface



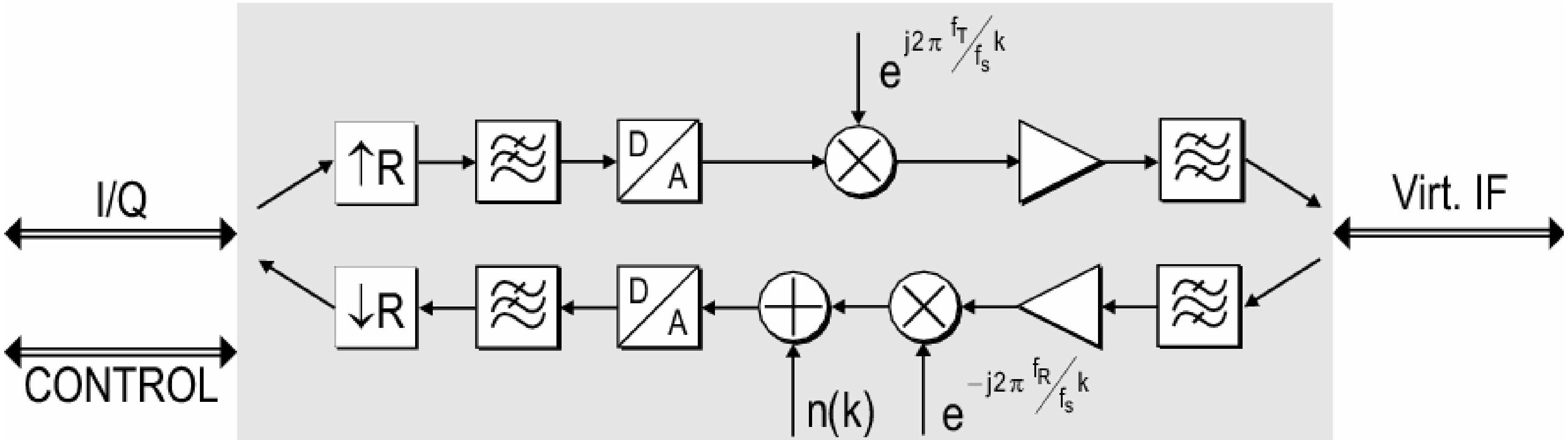
See Ettus Research LLC (http://ettus.com) and GNU Radio (http://gnuradio.org/trac/)

Implementation: Virt RF

Virtual RF front-end signal processing is a hierarchical GNU Radio block.

Virtual RF front-end

- Implemented as GNU Radio signal processing block
- Models internal USRP structure (DUC, DAC) and analogue processing
- Resulting digital signal is upsampled to simulation bandwidth (virtual IF)

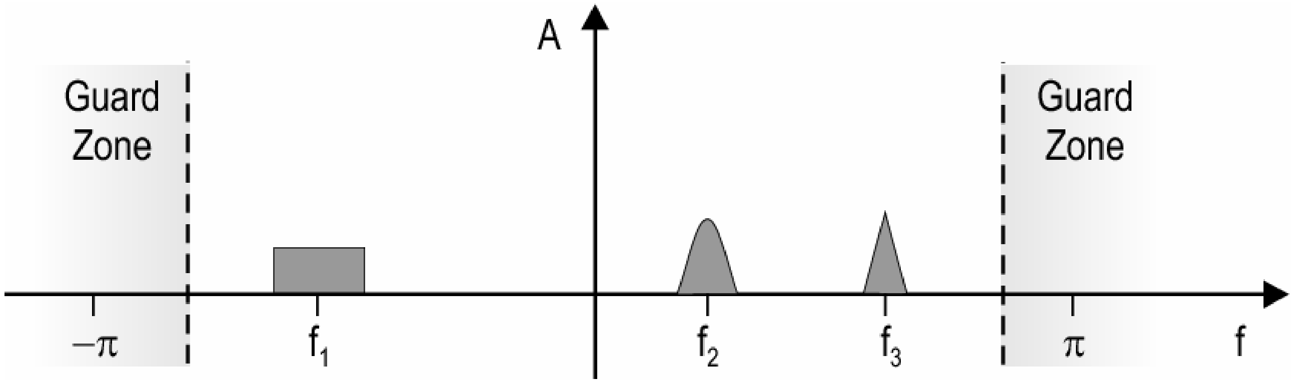


Implementation: Virt RF interpolation

Simulation bandwidth is shared by all SR terminals.

Virtual RF front-end and simulation bandwidth

- Virtual USRP interpolates to DAC rate, 128 MSamples/s
- Simulation bandwidth: 128 MHz
- Interpolation CIC/FIR-based, factor 4 – 512 (USRP1)
- I/Q base band rates from 250 kHz to 32 MHz
- Decimation accordingly as in USRP1



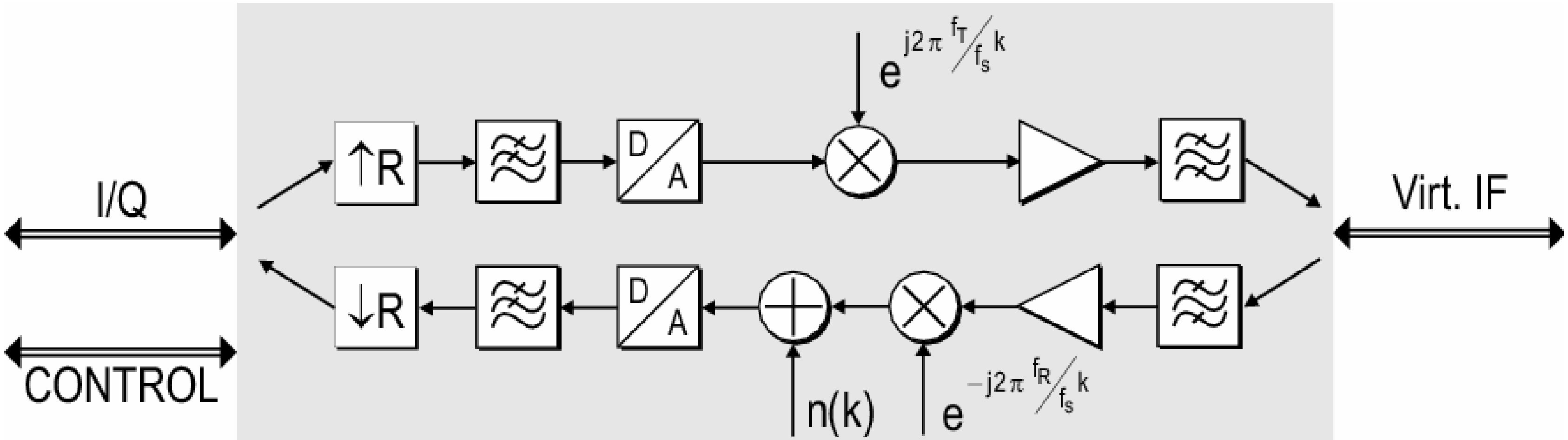
Simulation bandwidth with three SR terminals operating

Implementation: Virt RF modeled non-idealities

Virtual RF front-end models radio non-idealities .

Virtual RF front-end non-idealities

- Digital to Analog, Analog to Digital conversion: Quantization effects
- Frequency offset, drift
- Phase noise
- Non-linear amplification
- Analogue filtering
- Johnson noise

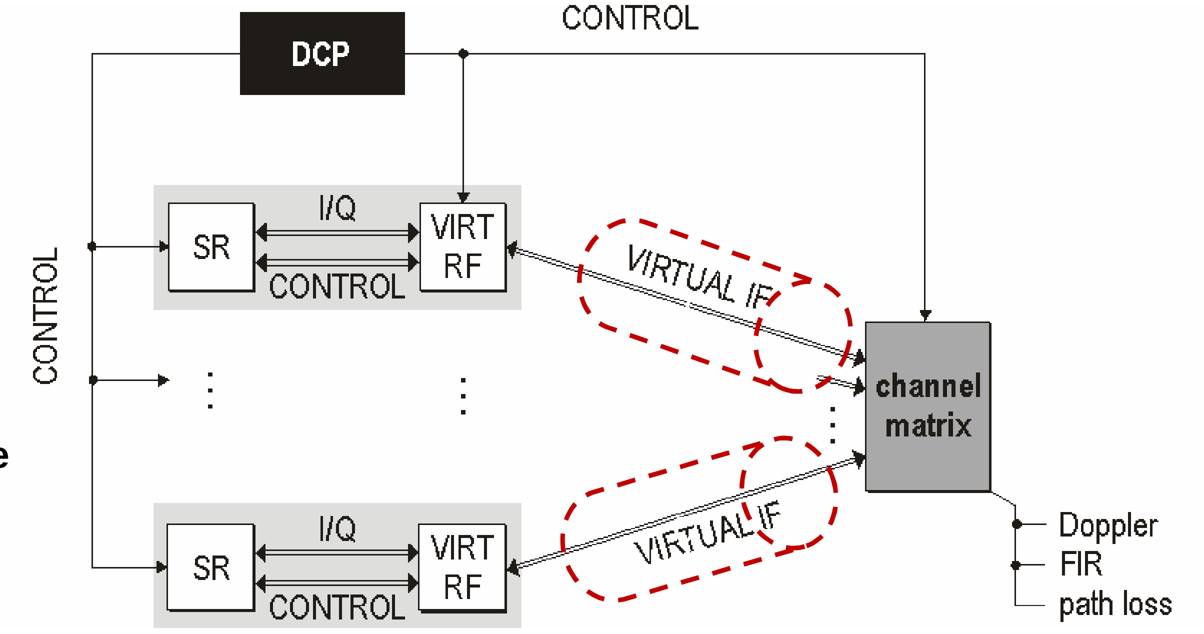


Implementation: Virt RF / channel API

SR terminal / Channel processes communicate via named pipes.

Data transfer between Virtual RF and Channel Matrix

- **Each SR terminal process interacts with channel matrix via named pipes**
  - FIFO buffer for inter-process communication
- **Channel matrix synchronizes streams on a sample per sample basis**
  - SR terminals provide streaming I/Q at simulation bandwidth



Implementation: Channel matrix

Channel matrix is hierarchical block: Time variant FIR tapped delay line model.

Channel Matrix

- The entirety of channels can between N SR terminals can be written in a NxN matrix of impulse responses
- Modeled effects:
  - Free space loss
  - Multi-path propagation
  - Doppler spread
- Resulting signal at terminal k is calculated via summation of filtered signals

$$\mathbf{H}_{\text{Chan}} = \begin{pmatrix} h_{1,1}(t,\tau) & \cdots & h_{1,N}(t,\tau) \\ h_{2,1}(t,\tau) & & \vdots \\ \vdots & \ddots & \vdots \\ h_{N,1}(t,\tau) & \cdots & h_{N,N}(t,\tau) \end{pmatrix}$$

$$r_k(t) = \sum_{l=1}^{N} s_l(t) * h_{l,k}(t,\tau)$$

CEL

Overview

Wireless Networks In-the-Loop: Software Radio as the Enabler

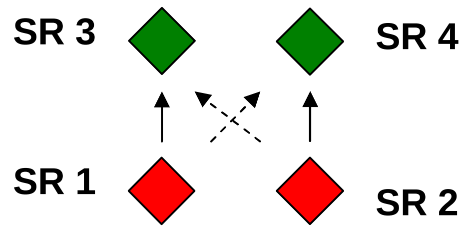
Sample Network: Checking consistency

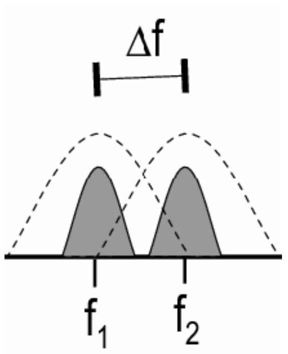
A simple SR network with 4 SR terminals is simulated.

Simulating an SR network

- Test consistency between Real Mode and Virtual Mode with a simple SR network setup
  - Virtual mode: simulation
  - Real mode: measurement
- Here: Realistic packet error rate analysis under co-channel interference for a complete SR terminal stack
- Transmitters: SR 1, SR 2, receivers: SR 3, SR 4
- Frequency flat fading assumed

Network setup





$$\mathbf{H}_{\mathrm{Chan}} = \begin{pmatrix} 0 & 0 & h_{1,3} & h_{1,4} \\ 0 & 0 & h_{2,3} & h_{2,4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

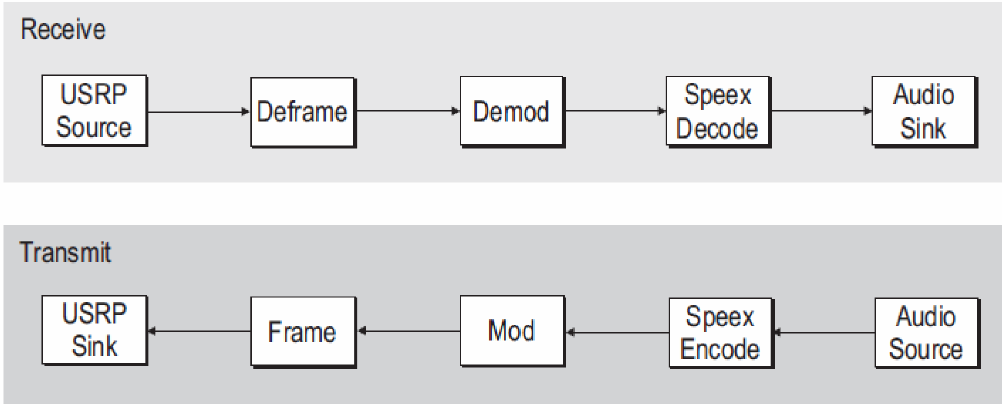
Sample Network: Checking consistency

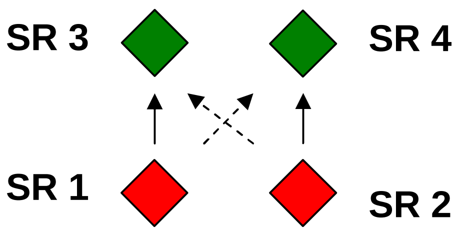
A simple SR network with 4 SR terminals is simulated.

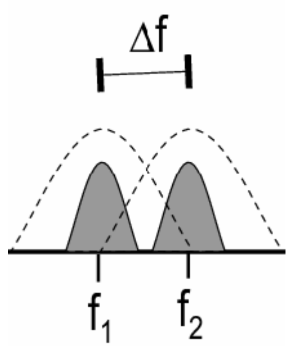
SR terminal code

- Flow graph for each terminal uses standard GNU Radio framing/deframing, modulation
  - GMSK on PHY
- Custom code for Speex
  - Narrow band codec for speech
  - Supports Packet-loss concealment
- No channel coding used

Flow graph







See Speex: A Free Codec for Speech, http://www.speex.org

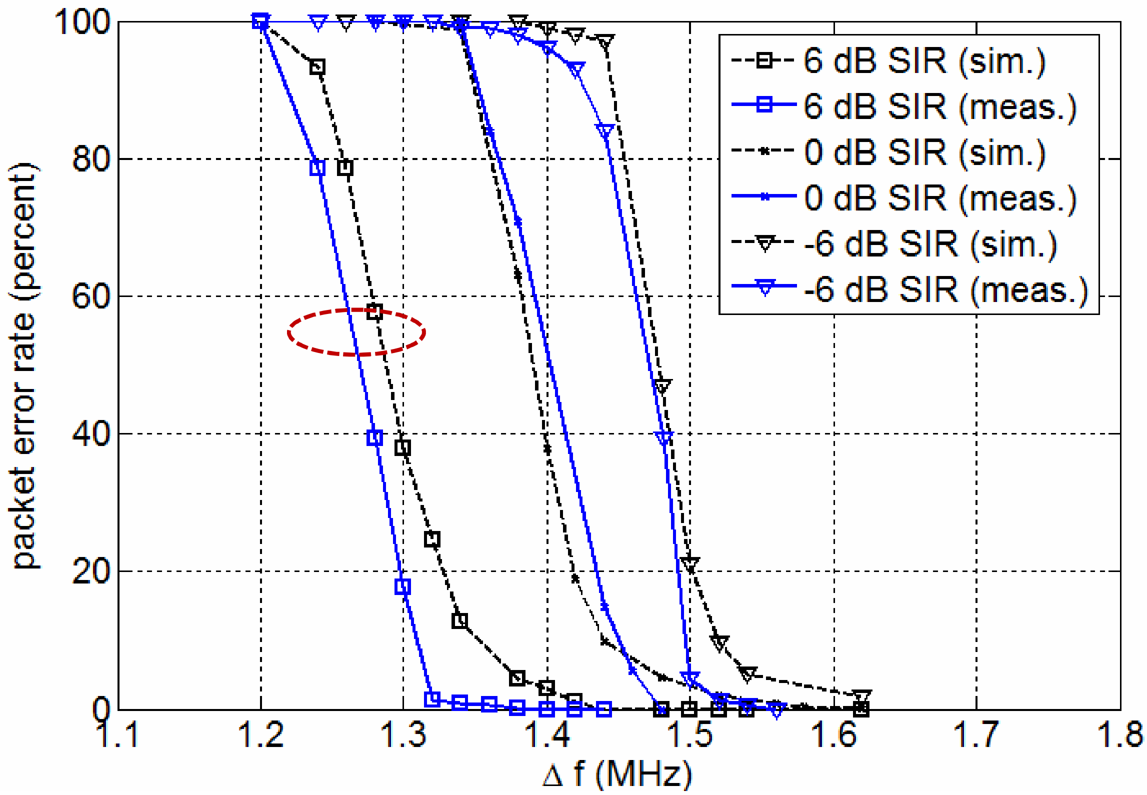
Sample Network: Checking consistency

A simple SR network with 4 SR terminals is verified by measurement.

Simulation and Measurement results

Modulation	: GMSK
Interpolation	: 64
Decimation	: 64
Samples/Symbol	: 2
Bit Rate	: 1 Mb/s
Tx1 SNR	: 22 dB
Tx2 SNR	: Varied
Signal BW	: 1.5MHz
RF Filter	: 2 MHz

SIR : Signal to Interference Ratio is calculated as Rx power of SR 1 to Rx power of SR 2 at SR 3



CEL

Overview

Wireless Networks In-the-Loop: Software Radio as the Enabler

- Introduction: Network Simulators and Software Radio
- Design Concepts
- Implementation
- Sample Network
- **Conclusion and Future Work**

Conclusion and Future Work

Wireless Networks In-the-Loop: On going work.

So far…

- Framework, sample implementation of a wireless network simulator, allowing loop-testing of wireless networks
  - Simple USRP abstraction
- Needed for verifiable performance testing of SR networks
  - Especially for ad hoc or overlay networks: evaluation of Interference
- Code under development

Goal: free network simulator

Future Work

- Introducing an absolute time reference
  - Sample-based synchronization is a kludge
- Improving performance
- Improving abstraction of hardware (e.g. accurate modeling of RF filters)
- Better demos, …, <your work here>

Q&A / Discussion

Thank you for your attention!