AN EXAMPLE OF WIRELESS DISTRIBUTED COMPUTING NETWORK ON 'CORNET'

Xuetao Chen (Wireless@VT, Virginia Tech, Blacksburg, VA, USA; chenxt@vt.edu); Tamal Bose (Department of ECE, University of Arizona, Tucson, AZ, USA; tbose@arizona.edu); Haris I. Volos (Department of ECE, University of Arizona, Tucson, AZ, USA; hvolos@arizona.edu); Joseph D. Gaeddert (Wireless@VT, Virginia Tech, Blacksburg, VA, USA; joseph.gaeddert@vt.edu); and Jeffrey H. Reed (Wireless@VT, Virginia Tech, Blacksburg, VA, USA; reedjh@vt.edu)

ABSTRACT

Wireless distributed computing networks (WDCNs) can group multiple wireless devices to fulfill a high performance computing task. It has several unique problems compared with wireless communication networks and traditional distributed computing networks. In this paper, we will introduce a design and implementation of an example WDCN developed on the Virginia Tech COgnitive Radio NEtwork Testbed (VT-CORNET). The paper first introduces the design and implementation of the software dedicated to WDC based on SDR technologies in order to demo the concepts and compare the experimented data with the theoretical results. It then presents the theoretical analysis of delay performance for proposed resource allocation methods for WDCNs. The results of experiments show that channel heterogeneity must be considered for WDCNs in order to reduce both the mean and variance of the execution time. The combination of SDR and multi-process programming makes it easy, based on this demo, to implement all kinds of WDCNs with different computing applications in the future.

1. INTRODUCTION

In this paper, we will introduce a demo of WDCNs developed on the Virginia Tech COgnitive Radio NEtwork Testbed (CORNET). Some of the proposed concepts in [1] can be demonstrated with the software developed in this paper.

CORNET is a collection of 48 radio nodes deployed throughout a research building on the Virginia Tech campus. All the radios are reconfigurable based on software defined radio (SDR) technologies. The architecture of CORNET is shown in Figure 1. There are 12 nodes on each floor. Each node consists of one RF front-end, USRP2, and a host server. Each USRP has a daughter board and a motherboard. The motherboard communicates with the host server through Ethernet. Each server has a Quadcore 2.13GHz CPU and 3GB RAM in order to support signal processing for the base band signal. These 48 servers form a server cluster through Ethernet connections and switches. The whole testbed is managed by five management servers which can provide the remote control and the access capabilities to the testbed through a secure shell from anywhere in the world.

The motherboard of USRP2 contains a Virtex Spartan3 FPGA. The receiver chain has a 14-bit analog-to-digital converter with 100 M samples/second speed and the RF chain has a 16-bit, 400 M samples/second digital-to-analog converter. This structure provides a capability of up to 50 *MHz* bandwidth with 16 bit/complex samples. Each motherboard can support up to two daughter boards. The current daughter board is a wide-band transceiver, which supports the frequency range from 50 *MHz* to 2.2 *GHz*. It has a maximum transmission power of 20 *dBm* and a noise figure of 5 *dB*. A custom designed daughter board using Motorola's RFIC is under development. RFIC is a multi-band direct conversion RF chip with 90 nm CMOS technology developed by Motorola Research Lab [2]. Its carrier is tunable from 100 *MHz* to 2 *GHz* with variable bandwidth of 10 kHz to 20 *MHz*. In the future, low-power mobile nodes will also be available.

Both GNU radio and Open Source SCA Implementation-Embedded (OSSIE) are installed in each host server. GNU radio is open source software for SDR development, which provides libraries for radio reconfiguration and control. OSSIE follows the Software Communication Architecture (SCA) developed for Joint Tactical Radio System (JTRS) for military software defined radios. The programs developed with these two are not generally compatible.

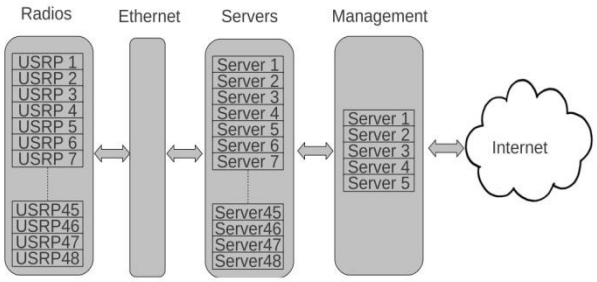

Figure 1 Software architecture for WDCN demo.

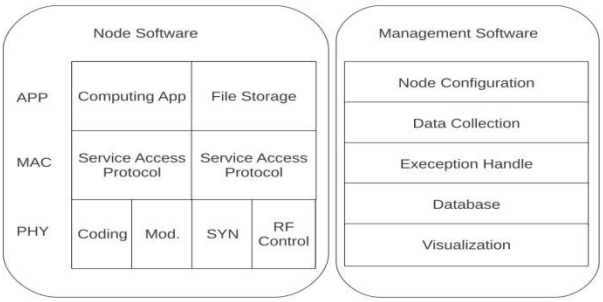

Figure 2: The system architecture of CORNET

All host servers and management servers are mounted in a 45U server rack cabinet located in a control room. CAT6

cables are used for the connection between the USRPs' Ethernet port and the servers' Ethernet port. It is easier to maintain and manage the servers in a centralized architect-

ture than in a distributed architecture. CORNET provides an open research and development platform for medium scale cognitive radio networks research such as distributed CR operation, CR network security, dynamic spectrum access and spectrum coordination. Detailed information about CORNET can be found in [3].

The demonstration of wireless distributed computing with CORNET has the flexibility that a network interface card (NIC) cannot provide. The radio parameters are reconfigurable and multiple PHY and MAC protocol may be developed with the same software tool. Different types of WDCNs can be demonstrated easily by replacing the computing applications. The server cluster has the potential for data mining. Mobile nodes may be integrated into the demo in order to study the impact of heterogeneous devices.

However, there is a limitation of the node structure currently used by CORNET. The digitized base band signal from USRPs is processed in the user domain of the host server. All the decisions and controls are done at the general purpose processor (GPP) of the host server. Although this structure has the best flexibility in terms of software/hardware reconfiguration, it also introduces latency and jitter issues [4]. It is highly possible that GPP is interrupted by other processes during its processing of USRP data. Kernel level delay measurements indicate a latency range from 600 μs and 15 *ms*, which is too high for a contention-base protocol. When the number of nodes increases, contention increases due to the high latency and jitters that leads to failures in decision-making and multiple access control (MAC). A possible solution is to split the functionalities of the MAC protocol between the USRP and host server, which requires programming FPGA and DSP on a USRP [4]. Another possible solution is to use a TDMA protocol with guarding time slots. In this demo, we will choose the second solution.

The computing task in this demo is a video compression with high compressing ratio, such as 40 times. A H.264 encoder is used for this purpose [5]. A service request node

(SRN) has a certain number of frames to be compressed. It has the option to compress by itself or distribute to service nodes through wireless links over USRPs. A SRN can distribute the video frames evenly without considering the channel condition or unevenly with consideration of the channel heterogeneity. These two strategies may have different impacts on the system delay and power efficiency. This demo shows that considering channel heterogeneity is the key to building power efficient and robust WDCNs.

Among the links between a SRN and SNs, some have an outage probability of zero and the others have an outage probability of p_o. This difference in outage probability is called channel heterogeneity in this paper. The workload allocation considering channel heterogeneity calculates the number of frames that each SN should compress according to the method proposed in [1]. This will reduce both the mean and the variance of the total computing time compared with the case frames are allocated evenly among SNs, which is the method without considering channel heterogeneity. The demo developed in this paper is expected to show this result.

Section two will introduce the software design and implementation. A demo using this software is presented in the following section, which includes a simple discussion of theoretical performance results and its comparison with experiment data. The detail information of theoretical analysis can be found in [1]. Then the last section summarizes the paper.

2. SOFTWARE DESIGN

The software architecture is shown in Figure 2. Each node runs node software, which integrates the communication process and computing process. The physical layer is based on the Liquid Radio DSP library [6] with OFDM capabilities. The reconfigurable parameters include: frequency, bandwidth, sub-carrier space, modulation, coding, transmission power, and packet length. The MAC protocol uses the TDMA protocol based on the polling and reporting mechanism. Service request nodes poll the status of service nodes and make decisions about workload allocation and transmission scheduling based on reports from the service nodes.

The MAC protocols for service access and service provision are different since service access does not require activating the computing process and accessing the file system.
During service provision, H.264 video compressing is the computing application. All the compressed data collected by a SRN can be de-compressed and compared with the original raw data. It is easy to replace video compression with other desired computing tasks as required. The node software is written in the C language. A UHD driver is required in order to control USRPs.

Management software is also implemented to coordinate the nodes configuration, data collection, and visualization. It is cumbersome to log in to every CORNET node and make the configuration. Therefore, the *Expect* language is used to automatically log in and configure multiple CORNET nodes. This network management software controls and monitors the nodes through Ethernet and is deployed on one control node while node software is deployed on each CORNET node.

The node software for each CORNET node can be divided into two parts: the software for service request nodes and the software for service nodes. The flow charts for these two are shown in Figure 3. During the service access phase, the SN will initialize the configurations of radios and create two processes, one for communication and the other for computing. The SRN will send out service request messages. Once it gets responses, the SRN will store the node ID of the SNs into a table and distribute the workload to SNs based on the proposed resource allocation method.

Then the service provision phase follows and the computing process on SNs starts compressing the raw data. The SRN can poll the status of the SNs periodically during service provision phase. The communication process on the SN responds to these inquiries with its own status. Once the data compression is finished, the SN will report to the SRN and request data transmission. The SRN will make a decision about the priority of the transmission based on its record about the status of the SN nodes.

Once it receives the compressed data from one SN node successfully, the SRN will remove the ID of this service

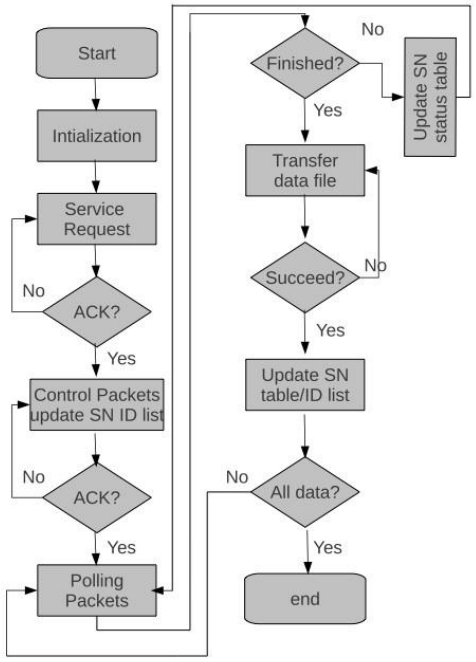

Figure 3: Control Flow chart for SRN.

node from its polling table and inquire the computing status of the rest of the nodes, which will speed up the response time. Once all the compressed data is received, the SRN will record the statistics of the experiment. It can also de-compress the raw data and play it back.

The node software encapsulates the service access and service provision into four functions: *wdc_control_tx*, *wdc_control_rx*, *wdc_txbuffer*, and *wdc_rxbuffer*. *wdc_control_tx* and *wdc_control_rx* provides the service access capabilities. The other two provide service provision capabilities. In this way, the performance of these two phases can be investigated independently.

Multi-process programming is used to implement the functions of the SNs. The communication process and computing process are affiliated to separate CPU cores in order to run independently without interfering with each other. The two processes can communicate with each other about their status using a signal library. When an exception occurs, the exception-handling module of the network management software will kill the process in order to re-install the normal operation. Network management software can also enable the debug mode of the node software, which helps protocol modification in the future.

A storage file can be used to decouple the communication process and computing process. The SN creates two processes responsible for communication and computing,

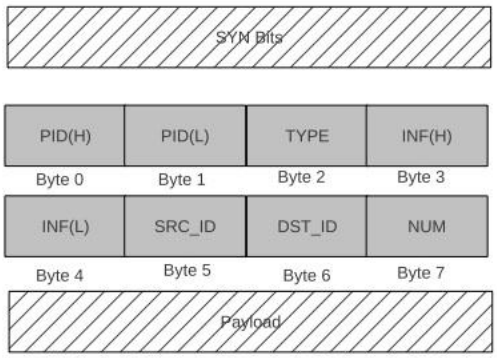

Figure 4: one packet for WDC demo4.

respectively. When the SN responds to the SRN, the most important status the SRN needs is whether the computing process is finished. Instead of using the signal library to update the status between the two processes, the existence of a compressed data file can be used as a control message between processes. The encoder writes the compressed data into a temporary file.

Once the encoder compresses all the raw data assigned to it, it will rename the data file with a new file name consisting of the node ID and a file suffix of "*snr*". The communication process checks the existence of this file in order to determine whether the computing process is finished. Although this may increase the delay compared with the implementation with a signal library, it has no impact on the communication process when replacing the video compressing application with any other computing applications, which makes it much easier to demo other WDCN implementation in the future.

Figure 4 shows the format for one packet. The first 128-bits are used for the purpose of synchronization. The packet header with 8 bytes is reserved for control information and protected with the strongest modulation and coding scheme. Source ID (SRC_ID) and Destination ID (DST_ID) are used to identify the wireless link. Packet Type (TYPE) has four values namely for the control packets, data packets, and their ACKs. The number of packets (NUM) and payload information (INF) help buffer allocation and decisions about the status of the data transmission. The payload length can vary according to different requirements.

3. A DEMO

One cluster with a SRN and multiple SNs is used to demo the concepts of WDC, especially the channel impact on the delay performance. Due to the limitation of latency and jitters, the real-time channel estimation is currently impossible. A channel emulator has been built into the software to demo the impact of channel heterogeneities on

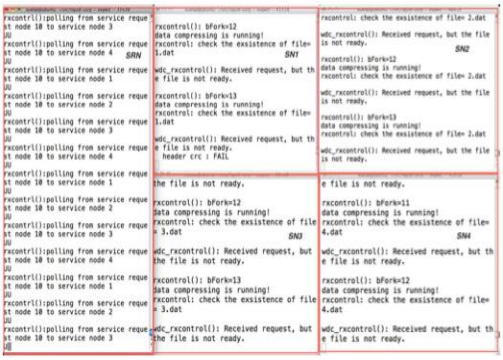

Figure 5: Debug mode of the WDC demo captured during the service provision phase

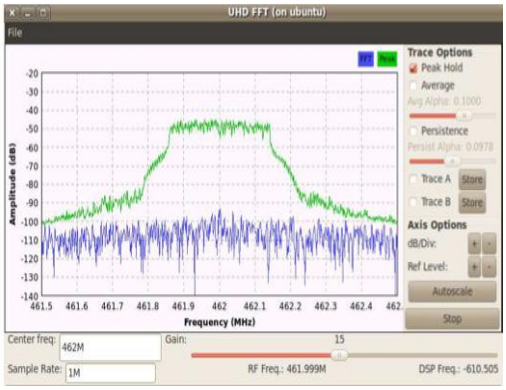

Figure 6: Over-the-air (OTA) waveform6.

the performance. Due to different outage probabilities, the number of re-transmissions may vary, which leads to a random communication delay. When the outage probabilities vary from link to link, the heterogeneous wireless channels are emulated.

3.1. Illustration

The debug mode is shown in Figure 5. This figure is captured during the service provision phase. The SRN, with *Node_ID* =10, on the left is polling the status of SNs according to its *SN_List*. SNs respond with their computing status, with $Node\_ID \hat{I} \{1,2,3,4\}$. The SNs windows throw the exceptions as needed, such as a cyclic redundancy check error for SN 1. Each SN has a counter to show the number of the received inquiries. Both the SRN and SNs

initialize timers at proper locations in the software in order to record communication delay and computing delay, which will be used to analyze the performance later.

Figure 6 shows the over-the-air waveform captured by another CORNET radio with a function from GNU radio [7] when the demo is running. The waveform has the shape of an OFDM signal with bandwidth of *B=300 kHz* and center frequency of f_c*=462MHz*. These RF parameters can be changed as needed. Currently, CORNET nodes support a frequency range from 50 *MHz* to 2.2 *GHz*.

It should be noted that the amplitude in Figure 6 is a relative one since all the RF chains of the USRPs are not calibrated. The daughter board needs to be calibrated as follows in order to have the absolute value of power in *dBm*. First, the thermal noise floor should be calculated and calibrated. Then the gain errors of the RF chain and thermal noise can be de-coupled with an injected reference RF signal, which will convert the unit of received power from *dBi* to *dBm*.

3.2. Workload Allocation Algorithm

Workload allocation methods considering channel heterogeneity follow the concepts in [1]. If there are *N* service nodes in a demo, half of them, $n \in S_h$, have bad channels with an outage probability of p_0 while the other half, $n \in S_\sigma$, have good channels with an outage probability of 0. For each service node, *n*, the compressing time $\tau(n, m)$ is a function of the number of frames, *m*. The communication time, $\tau(n, p_0, k)$ is a function of the outage probability p_0 and the number of packets *k*.

If each transmission costs T_s seconds, the average execution time $T(f)$ for each possible workload allocation scheme *f* can be represented as follows

$$T1(f) = \sum_{n \in S_b} \frac{N}{2(1-p_o)} T_s k(n) + \tau(n, m)$$

$$T(f) = \max\{T1(f), \tau(n, \frac{2M}{N} - m)\} + \sum_{n \in S_g} \frac{N}{2} T_s k(n)$$

The optimal workload allocation policy *f** can be found as

$$f^* = \arg\min_{f \in F} T(f)$$

where *F* is the set of workload allocation schemes. The reference workload allocation is evenly distributed as *m = M/n* frames to each SN.

3.3. An Experiment

In this demo, the workload is a video with *M =12* frames. SN 2 and SN 4 have an outage probability of zero while SN1 and SN3 have an outage probability of p_o. A larger p_o indicates greater difference between bad and good links, or larger channel heterogeneity for the whole network. The method considering channel heterogeneity allocates frames with equations in section 3.2. For the method without considering channel heterogeneity, *m=3* for all $n \in S$. This demo intends to show the impact of this channel heterogeneity on delay performance and workload allocation methods.

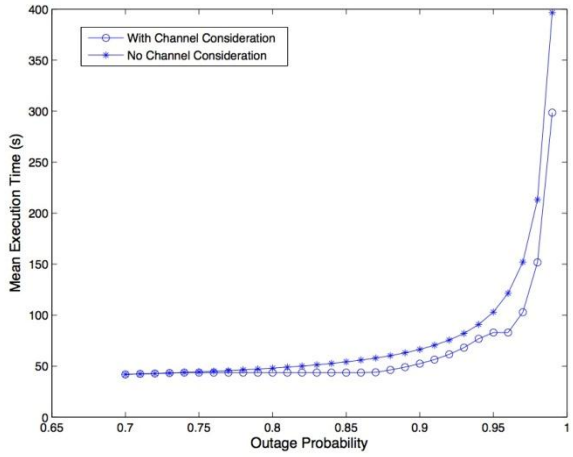

Figure 7: Estimated average delay performance as a function of outage prob. and workload allocation methods.

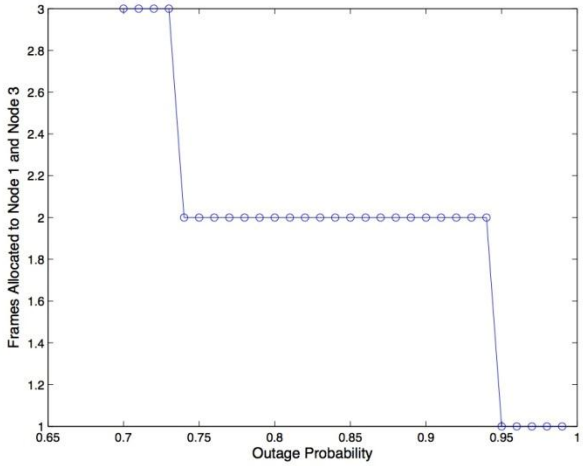

Figure 8: Optimal workload allocation as a function of outage probability.

Figure 7 shows the analysis results for average delay performance. It compares the results between the reference workload allocation method and the method considering channel heterogeneity calculated as equation in section 3.2. When p_o increases, the difference between good and bad links increases and will have more impact on delay performance. When p_o is smaller than a threshold around, $p_o = 0.75$, channel heterogeneity has no improvement on

delay performance. In other words, there is no need to consider the channel heterogeneity when $p_o \le 0.75$ in this demo setup. Figure 8 shows the optimal workload allocation for $n \in S_b$. *m=M/n=3* frames is the reference workload allocation scheme. When $p_o \le 0.75$, the method without considering channel heterogeneity. It has a staircase shape similar to the results in [1] since the minimum divisible unit for the workload is one frame in this demo.

Table 1. Experiment Data for Service Provision Time

	Reference Method		Proposed Method	
p_o	Mean	Variance	Mean	Variance
0.8	51.3	23.2	48.5	1.5
0.85	56.0	66.1	51.6	2.6
0.9	66.4	116.8	57.2	69.2
0.95	94.0	319.6	83.9	94.7

Table 1 show the experimented data collected from the demo on CORNET. They show the impact of channel heterogeneity on the mean and the variance of the execution time during the service provision phase. There are four test cases of SN 1 and SN3 having an outage probability of $p_o \in \{0.8, 0.85, 0.9, 0.95\}$, respectively. When the outage probabilities of SN1 and SN3 increase from 0.8 to 0.95, the channel heterogeneity increases. The method without considering channel heterogeneity distributes workload evenly among four SNs. The method with channel consideration allocates the frames according to the results in Figure 8. As expected, the method with channel consideration will reduce both the mean and the variance of the execution time. Both the mean and variance increase when the channel heterogeneity increases.

4. SUMMARY

An implementation and a demonstration of WDCNs are shown in this paper. Software dedicated to WDC based on SDR technologies is designed and implemented. The resource allocation ideas in [1] are demonstrated. The experimented data matches with the analysis results of the delay performance.

The results of our experiments also demonstrated that channel heterogeneity must be considered for WDCNs in order to reduce both the mean and variance of the execution time. The combination of SDR and multi-process programming makes it easy, based on this demo, to implement all kinds of WDCNs with different computing applications in the future.

5. ACKNOWLEDGEMENT

This work was supported by NSF PFI Grant No. 23708-0229-S02 and NSF Grant No. 1127953

6. REFERENCES

[1] X. Chen, *Resource Allocation for Wireless Distributed Computing Networks*, PhD dissertation, Virginia Polytechnic Institution and State University, April, 2012.

[2] G. Cafaro et al., "A 100 MHz2.5 GHz Direct Conversion CMOS Transceiver for SDR Applications, *in Proc. IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*, Honolulu, Hawaii, Jun. 2007.

[3] T. Newman, S.M.S Hasan, D. Depoy, T. Bose, and J. Reed, "Designing and deploying a building-wide cognitive radio networks," IEEE Communication Magazine, vol. 48, no. 9, Sept. 2010.

[4] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC pro- tocol implementations on software-defined radios," *NSDI 2009*, Berkeley, CA, 2009.

[5] H.264 Reference Software, online resource: http://iphome.hhi.de

[6] J. Gaeddert, Liquid Radio DSP library, online resource: www.ganymede.ece.vt.edu

[7] GNU Radio, online: http://gnuradio.org