CRI: CRD: A Web-Accessible Grid-Computing Resource for the Telecommunications Research Community

West Virginia University
Matthew C. Valenti, PI

Abstract: This project will provide researchers in the fields of information theory and communication theory with a grid-powered web service via a web portal interface. The web service will enable rapid execution of computationally demanding computer simulations that support the research of those working in the field. It is necessary because compute-intensive simulations are essential to progress in the field, yet the associated computational requirements often exceed the available resources of all but the largest institutions.

The extant grid supplying power to the initiative leverages the idle capacity of participating computers. Rather than deploying a dedicated infrastructure, it uses existing computers in the libraries, student labs, and staff offices of the participating institution and other participating institutions that chose to donate its idle resources. By eliminating the need to purchase dedicated machines, the cost of the initiative is greatly reduced and its self-sustainability does not depend on the ability to replace hardware in the years ahead.

The software to be run on the grid leverages an extensive free Matlab-based library developed by the Principle Investigator and used by an existing base of over 400 scientists and engineers. To simplify the user experience and obviate the need for users to purchase Matlab licenses, a web portal interface will be developed. Researchers from other institutions will be given access to the service and will be encouraged to donate their own idle computer capacity to the initiative.

Intellectual Merit: Research in the fields of communication theory and information theory is highly mathematical. The underlying mathematics often involves multidimensional integrals that can only be solved via Monte Carlo integration or simulation. Examples include the design of efficient error correction codes, the calculation of channel capacity under modulation constraints, and the design and analysis of wireless ad hoc networking protocols. The grid-based infrastructure that lies behind the proposed web service will allow rapid parallel execution of Monte Carlo simulations. The consequence is that researchers will obtain results faster (by orders of magnitude) than could be obtained when running on a single processor.

Broader Impacts: There are both research and educational broader impacts. The research impact is that the proposed web-based simulation service and underlying grid-based infrastructure will provide world-class computing power to groups that might not otherwise have access. The proposed account creation and management policies will allow for underrepresented or disadvantaged groups or institutions to have a larger share of computing resources. Dissemination will be accomplished by continuing to publish the underlying software under an open source license, by presenting tutorials on its usage at conferences in the field of communications and information theory, and by enhancing course content with exercises and projects that require the use of the proposed infrastructure.

1. Overview, Objectives, and Significance

Communications technology is a branch of information technology (IT) that has revolutionized the way that society interacts, works, and learns. Notable advances in communications technology over the past few decades include the cell phone, data modems (including telephone line, cable, and DSL), wireless computer networking (e.g. WiFi/802.11 and Bluetooth), and digital high-definition television (over cable and satellite). Communications technology has a positive economic impact on society. For instance, the base of 240 million wireless phone subscribers in the United States generates annual revenues of over $100 billion[1]. The communications technology industry is a vital place for young engineers to work, and will continue to demand the graduates of our academic institutions, provided that these students are prepared with the background necessary to make an impact.

Two trends in communication technology have been the focus of recent reports from the National Research Council. *Embedded, Everywhere* [NRC01] reports on the trend towards embedding IT into a wide range of networked devices, creating a type of digital nervous system enabling monitoring and control over a wide class of spaces, including environmental monitoring, highway traffic control, precision agriculture, scientific monitoring, biotechnical research, homeland security, and battlespace surveillance. The ability to efficiently connect a very large number of autonomous devices is the main technical challenge cited in the report, and the solution is likely to involve advanced wireless technology. Since the report was published, Zigbee (IEEE 802.15.4) technology has emerged as a viable candidate for low-cost networking of embedded devices. *Broadband: Bringing Home the Bits* [NRC02] focuses on the challenges associated with last-mile high speed data connectivity, primarily to the home, and identifies fixed wireless (now known as WiMax) as a particularly attractive option due to its scalability and ability to be rapidly deployed. As an example of the high stakes involved, on July 5, 2006, Intel announced a $600 million investment in *Clearwire*, a WiMax service provider whose business plan hinges on providing wireless last-mile broadband connectivity [Ed06].

Given the societal and economic benefits of communications technology, it is no surprise that it is the focus of an enormous amount of research effort and is a popular area of specialization among electrical engineering, computer engineering, and computer science students. Traditionally, the barriers to entering the communications research community have been rather low. Work during the second half of the 20th century tended to be analytical in nature, and small research groups could build simple proof-of-concept hardware experiments on a minimal budget. Furthermore, since the hardware was simple, it could be simulated on a standard desktop computer using accepted mathematical models for the communications channel.

Over the past decade, the complexion of the communications research community has begun to change. Systems have grown so sophisticated, that often a closed-form analysis does not exist and analytical work can only provide performance *bounds* that are often quite loose. Accurate predictions of performance must now rely on computer simulation. However, because the systems are so complex, even an efficiently written simulation will take an unacceptably long time to complete on a single processor system. *Compute power* has become the differentiator amongst research groups, and a new digital divide is emerging in the communications research community. Well-funded groups have the ability to purchase cluster computers or compute time for the purpose of running simulations, while smaller groups with less funding are relegated to running simulations on a handful of processors. As an example, a particular low-density parity-check (LDPC) code[2] was developed by a research group in Japan for consideration in the

[1] http://www.ctia.org

[2] LDPC codes are given as an example in section 3.1

10GBase-T Ethernet standard and simulated on a 256 node supercomputer for two weeks [Ka04]. The simulation allowed a bit error rate (BER) of 10^{-12} to be attained, and it was only at this low error rate that the benefits of their proposed code became apparent. The same work would have taken over 9 years running on a single 3 GHz PC computer. A small research group starting at the same time as the Japanese group would still be running simulations. While the situation is alleviated to a certain extent by the availability of public supercomputing resources, such as the NSF supercomputing centers at Cornell, Illinois, Pittsburgh, and San Diego, the barriers to using these resources are too high for it to have made a significant impact on the wider communications research community.

We propose the creation of a new simulation resource for the communications research community with the goal of breaking down the aforementioned digital divide. Our objectives for this resource are for it to be:

(1) Open to the entire research community.

(2) Easy to use through the existence of a web interface.

(3) Able to access significant processing power at low cost by leveraging existing systems.

(4) Powered by simulation logic that is open, reviewable, and locally executable through the use of free software.

(5) Able to integrate contributions from the research community that extend functionality

(6) Sustainable through a credit system that rewards suppliers of CPU cycles and allows industrial participation through the purchase of cycles.

(7) Integrated into courses on coding theory, information theory, and wireless networking.

The proposing organization has built a proof-of-concept system that allows simulations written in Matlab to run on a grid of 30 machines. The proposed work will build upon this success by increasing the size of the grid and making it available to a much wider audience.

Our vision is to allow a researcher or student to access the resource by logging into a web portal. Once authenticated, the user could initiate a new simulation, check on the status of existing jobs, or download results from completed simulations. The simulations are set up within the web browser by selecting from a set of menus and, in some cases, uploading certain types of data files. The system will initially support three broad classes of simulations: (1) Error rate of coded modulation, (2) Capacity limits under modulation constraints, and (3) Performance of multi-terminal wireless networks.

Once a job is launched, it will be run on a computational grid, the nodes of which we will manage, though we will not physically maintain all the computers that comprise the grid. In contrast with a cluster computer, which has processors dedicated to specific computing tasks, our grid computer will primarily make use of existing computers, for instance in teaching laboratories, staff offices, and libraries. The grid computer makes use of idle CPU cycles on these machines, essentially creating instant infrastructure without the need to purchase dedicated machines for this project (except for a small cluster that will be used for testing and to guarantee a minimal level of service). Power users of this service will be expected to contribute machines from their own institutions to the project. Our goal is to increase an initial base of 100-200 machines at WVU to 500-1000 machines throughout the research community by the end of the four year project. Long term sustainability of the project will be ensured at the completion of the NSF funding cycle through a credit system whereby suppliers of CPU cycles are given positive credit, while users of the system have credit deducted.

The remainder of this proposal is organized as follows. A high level overview of how the system works will be given from the user perspective in section 2. In section 3, details of the three types of simulations that can be executed are presented. Section 4 provides architectural details of how the grid computing system will be used to provide simulation service, focusing on the specific plan of work and management plan. A plan for the implementation of the infrastructure is given in Section 5. A description of the educational use of the proposed system is given in Section 6, while Section 7 discusses dissemination plans. Section 8 is a timeline, while Section 9 gives a summary of the broader impacts.

2. Infrastructure Overview: The User Perspective

In this section, we give an overview of how the user will interact with the system. The user might be a graduate or undergraduate student who needs to access the system to complete a course project (see Section 6) or to support his or her thesis/dissertation research. Alternatively, the user might be research faculty at an educational institution, perhaps one that does not otherwise have extensive computing power. Upon opening an account, the user will be given credit for a certain number of CPU hours. While all users will be given a certain base number of hours (e.g. 500), users from certain institutions, including Historically Black Colleges and Universities (HBCU), Hispanic Serving Institutions (HSI), Tribal Colleges and Universities (TCU), other Minority Institutions (MI), and universities from EPSCoR states will be granted significantly more CPU hours (e.g. 4,000). Knowing that industry is a big participant in the communications technology research community, policies will be in place towards the end of the NSF funding cycle that will allow industrial participation. In fact, it is the industrial participation that will assure the longevity of the project after NSF involvement has concluded.

The researcher will point his or her browser to a secure website. Once logged in, he or she will be allowed to start a new simulation, manage an existing simulation, or download results. Initially, there will be three broad categories of simulations from which to choose. The details of each of these types of simulations are presented in Section 3. Simulations are configured by selecting options such as the range of signal-to-noise ratios to simulate, the number of errors to log, the type of modulation, the channel type, and the type of error correction code. Some parameters allow the researcher to upload their own design, such as a parity check matrix for an LDPC code (discussed in Section 3.1) or the signal constellation for modulation-constrained capacity calculations (discussed in Section 3.2). Once the simulation parameters are specified, a button is clicked and the simulation started.

If a single user initiates a short simulation, it could be run locally on the same machine that hosts the web server. But this is not the usage that we envision. Instead, there will be many users who would like to each run several compute intensive simulations. To accommodate this large number of simulations, each one will be sent by the server to another computer on the grid running a *compute engine*. This computer might be an idle computer in a student teaching lab, it might be a computer in a laboratory, or it might be an office computer sitting on somebody's desk. From the researcher's perspective, it does not matter where it runs. Similarly, the compute engine does not negatively impact the person who is working locally on the machine hosting the compute engine. If it is a unix/linux machine, the simulation is given low priority (using the *nice* command), and if on a windows-based machine, it operates as a screen saver that starts only after a certain amount of local user inactivity.

After the simulation has been started, the researcher will typically log off and let the grid take care of details such as load balancing and shifting. After a sufficient amount of time (which depends on the complexity of the simulation), the researcher will log back in and check on the status of the job. The webpage will list the status as queued, running, paused, or completed.

Regardless of the status, the user can download the current state of the simulation, which gets reported back from the compute engine to the server every 5 minutes. Using local software, such as matlab or a plotting application that we will develop, the results can be plotted and incorporated into technical papers and presentations.

As a courtesy, the results from all past simulations are kept in the user's account, until they are deleted or a quota on user storage space is reached. Results can be kept private, or if the user elects, may be made available to the entire research community. In the case that a result is made available to the research community, the system will check to see if results exist prior to launching each simulation job.

The simulation software is based upon the Coded Modulation Library (CML), developed by the Principle Investigator[3]. This is a free software project that can run in matlab and already supports the simulation types described in Section 3 and has been executed on the 30 node proof-of-concept grid. The benefit of basing the simulation on free software is that the user can inspect the source code and does not have to pay to license the software. The research community will be encouraged to participate in the evolution of this software package by contributing additional features (e.g. new codes, modulations, or channel models) to the project. Throughout the project, the PI will serve as lead software integrator, with the responsibility of integrating contributions from the research community.

Within the web-interface, the researcher will see a balance of the number of CPU hours consumed and the number remaining. Once the number of CPU hours has expired, the user has two options if he or she wished to continue participation. On the one hand, the user can contribute his or her own machines (or machines from their department) to the grid. Alternatively, users may purchase CPU hours, however this option will not be available until close to the end of NSF funding cycle.

3. Research Projects and Preliminary Results

This section describes three example research projects that will benefit from the proposed infrastructure. It should be emphasized that the key feature of the proposed service is its methodology for seamlessly executing large-scale scientific computing jobs on a computational grid. While we initially anticipate an audience in the communications and information theory research communities, the architecture is extensible such that a larger set of researchers from all fields of computing, signal processing and mathematics could participate.

3.1 Simulation of Coded Modulation

Forward error correction (FEC) coding is a critical part of every modern communication system. FEC codes allow the system to recover from errors that are inevitably introduced when signaling over inherently unreliable mediums, such as wireless channels, coaxial cable, and magnetic recording channels. When properly leveraged, FEC coding allows the system designer to intentionally operate in a regime that produces more errors, knowing that the code can correct a significant number of errors. The ability to operate at low signal-to-noise ratios (SNRs) allows systems that can transmit over longer distances, transmit at lower power (and hence, have longer battery life), use smaller antennas, or allow more users to simultaneously transmit over the same channel (as in multi-terminal wireless networks).

[3] The software may be obtained at http://www.iterativesolutions.com/matlab.htm

There has been sustained research in interest in FEC coding ever since 1950, when Richard Hamming introduced the first FEC code that now bears his name [Ha50]. The period from 1950 to 1993 experienced slow but steady progress in coding theory with each new code offering advantages in efficiency and error correcting capability. A revolution in coding theory occurred in 1993 with the introduction of turbo codes [BeGl93], which were able to perform within a fraction of a decibel from the Shannon capacity limit (which is discussed in Section 3.2). Since then, turbo codes have become adopted by several commercial systems, including both third generation cellular systems (UMTS [ETSI05] and cdma2000 [3G02]), WiMax fixed broadband wireless (IEEE 802.16) and the DVB-RCS system for delivering Internet service via satellite [ETSI05]. The interest in turbo codes caused a renewed interest in low density parity check (LDPC) codes, a related coding technique invented by Gallager in 1960 [Ga60] but thought to be impractical until only recently. Industry has now begun to favor LDPC codes over turbo codes for certain applications, with LDPC codes being used for second generation digital video broadcasting (DVB-S2) [ETSI03] and the new mobile version of WiMax (IEE 802.16e) [IEEE05].

LDPC and turbo codes attract a large amount of research interest. For instance, at the 2006 *IEEE International Symposium on Information Theory (ISIT)*, the premier conference devoted to information theory, there were 17 sessions devoted specifically to LDPC codes (with about 4 papers in each session), more sessions than any other single keyword. Another 22 sessions were associated with other aspects of channel coding, including turbo, convolutional, and Reed Solomon codes. Similarly an inspection of journals such as *IEEE Transactions on Information Theory*, *IEEE Transactions on Communications*, and *IEEE Transactions on Wireless Communications* demonstrates the attention the research community pays to error correction coding.

Turbo and LDPC codes are both block codes whose encoding operation can be represented as the following vector-matrix multiplication operation:

$$\mathbf{c} = \mathbf{u}\,G \tag{1}$$

where **u** is a length k message vector, G is a k by n generator matrix, and **c** is a length n code word. The ratio of k to n is called the code rate r=k/n and is a measure of the spectral efficiency of the code. Generally, **u**, G, and **c** are binary, i.e. contain only the elements 0 and 1, but more generally, the codes could be non-binary (which is the case for Reed Solomon codes). Block codes can also be specified by their parity check matrix H, a matrix with n columns and n-k linearly independent rows (and possibly more linearly dependent rows) for which $\mathbf{c}H^T$ is equal to the all zeros vector for all valid codewords **c**, and is nonzero if **c** is not a valid codeword. LDPC codes are characterized by having low density parity check matrices (H matrices with many more zeros than ones), and it is the H matrix itself that defines the code.

The simulated performance of LDPC codes used by the DVB-S2 standard is shown in Fig. 1. The simulation used the CML toolbox and ran locally within matlab. Three code rates were simulated along with two code word lengths, long (n=64,800) and short (n=16,200). The simulation of the rate for ¼ codes took about two months to produce by running on a single machine.

With the proposed infrastructure, users will be able to set up simulations similar to the one used to produce Fig. 1. However, they will be given much more flexibility in their design choices. In Fig. 1, the modulation is limited to binary phase shift keying (BPSK), but in the proposed infrastructure, users can select from a wide range of modulation formats or can even define their own custom modulation. The results in Fig. 1 are for a specific LDPC code from the DVB-S2 standard. While users could use this code, they can also define their own LDPC code and upload the parity check matrix to the web portal. Other codes will also be supported. In addition, channels other than additive white Gaussian noise (AWGN) can be simulated, including Rayleigh fading channels (both block fading and ergodic fading).

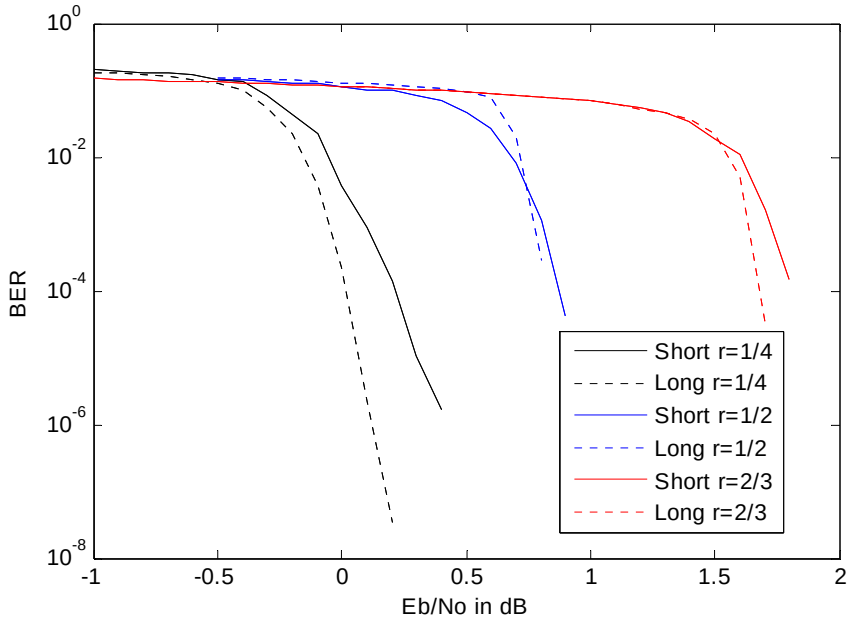

Fig. 1. Bit error rate (BER) performance of the LDPC code used in the DVB-S2 standard using BPSK modulation over an AWGN channel. Eb/No is the signal-to noise ratio.

In order to set up a simulation, the user will login to the web portal and will be allowed to pick the modulation, code, and channel from a list of alternatives. The list will contain the following options:

1. Modulation format: BPSK, QPSK, QAM, FSK, or user-defined (requiring an upload).
2. Demodulator type: Specifies how M-ary symbol likelihoods are converted to bit log-likelihoods at the receiver.
3. Code type: Turbo, LDPC, convolutional, Reed Solomon. For each code type the user can specify the code (through uploading a file) or can pick from industry-accepted standards like DVB-S2, UMTS, cdma2000, and DVB-RCS.
4. Decoding algorithm: Chose from a set of alternatives for each code type.
5. Channel type: AWGN, ergodic Rayleigh fading, block Rayleigh fading.
6. Range of signal-to-noise ratio (SNR) to simulate.
7. Number of trials or errors required for each SNR point.

The number of combinations that can be tested using this interface are unlimited. The rapid execution of the simulations encourages researchers and students to experiment with their designs, and through brute force trial-and-error, we anticipate that new combinations of coding and modulation will be discovered that are more efficient than what is presently known.

3.2 Calculation of Modulation Constrained Capacity

From Fig. 1, it is apparent that as the rate of the code gets lower and the length of the code gets longer, the system exhibits better performance (in the sense that a low error probability can be attained at a lower SNR). One might wonder what limits exist in the tradeoff between code rate, code length, and required SNR. This tradeoff is set by the *channel capacity*, which was initially derived by Claude Shannon in 1948 [Sh48] and forms the basis for the field of *information theory*. When one places no restrictions on the channel input (aside from an average power limitation), the channel capacity is given by:

$$C = W \log(1+SNR) \qquad (2)$$

Where W is the bandwidth of the channel, SNR is the signal to noise ratio, and if the log is taken to the base-2, then C has units of bits per second. C is the maximum data rate that can be supported over the channel at arbitrarily low error rate. When appropriately normalized for the choice of modulation, C determines the required code rate when operating at that particular SNR. Interpreted from another perspective, the relationship specifies the minimum SNR required as a function of code rate.

Equation (2) is derived without any restrictions on the choice of modulation. If one were to restrict the modulation to use a certain finite set of symbols, the expression for capacity changes. For a channel with input symbol X (drawn from signal constellation S) and output symbol Y, the *mutual information random variable* is defined as [Ga68]:

$$i(X,Y) = \frac{f_{XY}(X,Y)}{f_X(X)\,f_Y(Y)} \qquad (3)$$

where $f_{XY}(x,y)$ is the joint probability density function (pdf) of X and Y, $f_X(x)$ is the marginal pdf of X, and $f_Y(y)$ is the marginal pdf of Y. The capacity of the system constrained to use signal constellation S is found from the expected value of i(X,Y),

$$C = E[\, i(X,Y) \,] \qquad (4)$$

where the expectation is performed with respect to all realizations of X and Y. While in some specific cases, (4) could be found analytically, usually direct calculation is not possible or requires numerical integration over a large number of dimensions. A more feasible method for evaluating the capacity is to use Monte Carlo integration [CaTa98, MaCo98]. With the Monte Carlo method, a signal X is selected at random and sent over a random channel with simulated noise and (possibly) fading. The simulated channel output Y is used together with X and (3) to come up with a realization of the mutual information random variable. The process is repeated and the mutual information averaged until a desired level of confidence is achieved.

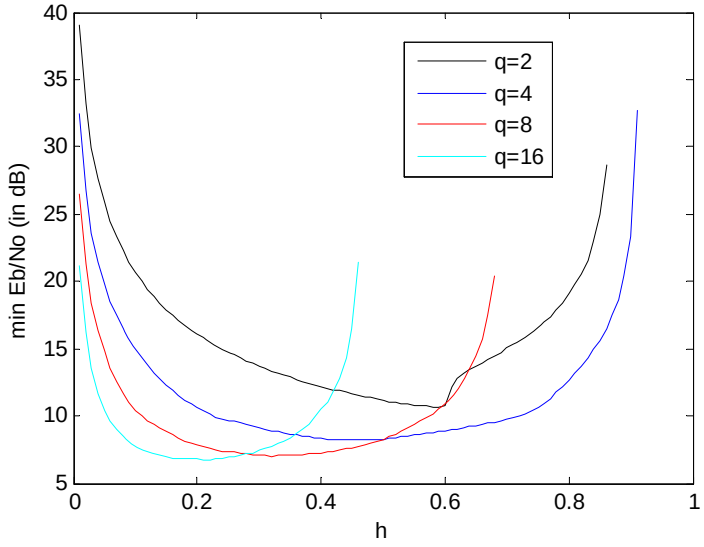

Fig. 2: Minimum SNR required to achieve arbitrarily low BER in a coded FSK system as a function of modulation index h over Rayleigh fading.

By using Monte Carlo techniques, the capacity of very sophisticated modulation formats can be determined. An example is shown in Fig. 2, which shows the information theoretic minimum SNR required for frequency shift keying (FSK). With FSK modulation, the transmitter selects from a set of q tones. The frequency separation between tones is determined by the modulation index h, with lower values of h resulting in more closely spaced tones. As h gets smaller the system becomes more spectrally efficient but performance degrades due to interference between adjacent tones. The optimal value of h to be used for a coded FSK system with a bandwidth restriction has been an open problem, but one that can be solved with the assistance of the proposed grid computing infrastructure.

The results show in Fig. 2 were generated using the preliminary implementation of the proposed grid computing system. For each value of q and h, a Monte Carlo simulation was run that determined the minimum SNR required as a function of code rate r. This required about 300 simulations to be run, and each simulation took about 8 hours to complete. If we ran on a single machine, it would have taken 100 days, but because we could run in parallel on 30 machines, we obtained the results in less than 4 days. From each simulation, the minimum SNR was determined and the result plotted in Fig. 2. Taking the optimization one step further, the optimal value of h for each value of q can be found by taking its minimum value shown in Fig. 2. A full explanation of this optimization and its application to FH systems has appeared in conference form in [ToCh07] and has been submitted to *IEEE Transactions on Communications* [ToCh06].

The proposed infrastructure can be used to determine the channel capacity of a wide range of systems under various constraints. A user can request that constrained channel capacity be calculated by selecting the following options within the web portal:

1. Modulation format: BPSK, QPSK, QAM, FSK, or user-defined (requiring an upload).
2. Demodulator type: Specifies how M-ary symbol likelihoods are converted to bit log-likelihoods at the receiver.
3. Channel type: AWGN or ergodic Rayleigh fading.
4. Range of signal-to-noise ratio (SNR) to simulate.
5. Number of trials for each SNR point.

Note that many of the options are similar to the choices for coded modulation simulations. The key difference is that capacity simulations do not require the channel code to be simulated, and therefore can execute much more quickly. This allows for optimization over a much wider parameter space than if the entire coded system had to be simulated.

We would envision that a researcher would first use this tool to determine the channel capacity of their selected modulation and demodulator type over the channel of interest. By iterating through several alternatives, a design will be selected for further study. At that point, the system can be simulated with an actual error correcting code (per section 3.1) in order to determine the actual performance of the coded modulation.

3.3 Wireless Ad Hoc Networks

While the focus of the first two research projects is point-to-point communications between a pair of terminals, practical wireless systems must operate in the presence of other terminals. Although the existence of other terminals is a disadvantage due to an increase in interference, it also presents opportunities to improve performance via multihop routing. In a multihop system, a message source can communicate to a destination located beyond immediate radio range by hopping through intermediate nodes that act as *relays* [Pe01]. Because radio transmissions are inherently broadcast oriented, every node within radio reception range will be able to decode the source's transmission provided that the signal-to-interference ratio (SINR) is sufficiently high. In particular, the SINR at a receiver is [Go05]:

$$SINR = \frac{P_0}{N + \sum_{i=1}^{n} P_i} \qquad (5)$$

where Where N is the total noise power, P_0 is the received power of the desired signal, and P_i the received power of the i^{th} interfering signal. The message will be decoded with high probability provided that C > R, where C is the channel capacity, found by substituting the SINR into equation (2), and R is the data transmission rate. Once a node has successfully decoded a message, it can serve as a relay and forward the message. If no node decodes the message, the source could retransmit using a hybrid automatic repeat request (ARQ) protocol [CaTu01]. Through a combination of retransmissions by the source and the relays, the message will progress towards the destination until it is eventually able to decode.

The performance of an ad hoc network can be described with respect to the tradeoff between energy, throughput, and latency. An example result showing the tradeoff between latency and energy efficiency is shown in Fig. 3, which is excerpted from our paper [ZhVa05]. The Figure compares performance of conventional multihop and a protocol we developed called *hybrid automatic repeat based intra-cluster geographic relaying (HARBINGER).* In conventional multihop, the message must pass through a pre-selected route comprised of point-to-point links. On the other hand, with HARBINGER any node that correctly decodes the message can retransmit it. HARBINGER bypasses bottlenecks by not requiring nodes that are experiencing deep fades or high interference to decode the message.

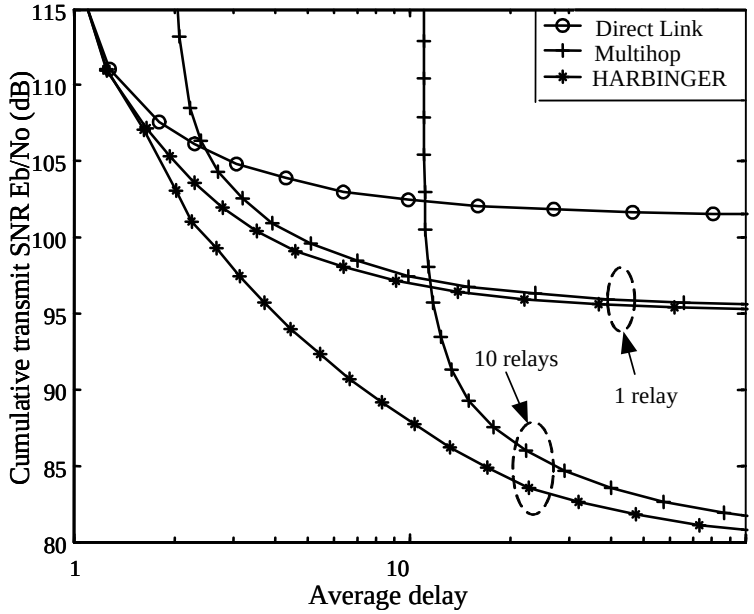

Fig. 3: Tradeoff between SNR and delay in an ad hoc network with 1 or 10 relay nodes. Figure compares performance of conventional multihop with the HARBINGER protocol of [ZhVa05].

In the proposed infrastructure, we will include support for the simulation of ad hoc networks. The user will be allowed to configure the simulation by picking the following parameters:

1. Network topology: Can be fixed (requiring a file upload) or random (requiring parameters such as node density to be entered).
2. Protocol specification: Protocols such as conventional multihop and HARBINGER can be selected. Users can specify their own protocols by describing behavior of the nodes.
3. Channel model: The model could be AWGN or block fading. The path loss coefficient can be specified by the user.
4. Transmit power and noise spectral density.
5. The code/transmission rate used by each user.

4. Description of the Infrastructure and its Alternatives

This section discusses the two main components to the proposed infrastructure, the simulation software and the grid computing architecture. For each component, we survey the available alternatives, and provide justification for the choices we have made.

4.1 Simulation Software

The simulation software is the science behind the infrastructure, and is what distinguishes the proposed project from a general-purpose computational grid. The software must enable all the research projects discussed in Section 3, yet be extensible to support future functionality contributed by the user base. Researchers should be able to have access to the simulation's source code to remove any doubt regarding how the simulations are implemented. Users should be able to run the simulation software not only on the grid, but also locally on their own PC computer. Support for local execution allows designs to be tested locally before they are sent to the grid, thereby conserving the researcher's quota of CPU hours. To reach the broadest possible audience, the software should ideally not require the purchase of licenses by the end user.

In surveying the alternatives, one cannot ignore that Matlab is the de-facto standard programming language within the communications research community. While not necessarily free software, most academic institutions have site licenses and the student version is fairly inexpensive. Matlab is a rich development environment. Developers can use Matlab to create graphical user interfaces, and by using the *Matlab Web Server*, applications can be accessed via the web. By using the *Matlab Compiler*, projects can be compiled into stand-alone applications, which can be distributed and run without needing a Matlab license.

With these benefits in mind, we propose using Matlab as our *development* environment. We have already used Matlab to develop most of the functionality required to enable the research projects discussed in Section 3. This software project, called the *Coded Modulation Library*, is available as free software under the Lesser Gnu Public License. The CML project has been publicly available since October 2005, and now has a registered population of over 400 users[4]. While Matlab scripts are used to implement high-level control of the simulation, lower level algorithms (e.g. the LDPC decoder) are implemented in C and called from Matlab as *c-mex* functions. The use of c-mex dramatically increases the speed and efficiency of the simulation.

In addition to distributing the Matlab and c-mex source code, which can be run on any machine with a valid Matlab license, we have also used the Matlab compiler to produce a stand-alone version of the simulator. Compiled code does not require a Matlab license to run, and so making the compiled code available widens the audience of the project to users that do not have Matlab licenses. One disadvantage of using the compiled code is that the user cannot change or add

[4] Because registration is optional, the actual user population is likely to be significantly higher.

functionality by modifying source code. However, users that want to change functionality without purchasing a Matlab license could run the software within Octave, a free alternative to Matlab[5].

The compiled Matlab code is a critical component in the grid computing architecture. When a job is run on the grid, the compiled code is sent to the compute engine. A key benefit of using compiled Matlab code is that the compute engine does not need to have its own Matlab license, though it does need to be preloaded with the Matlab Runtime Environment, which takes up about 300 Megabytes of disk space. In Section 5.3 we propose a workaround to this requirement.

When making the decision to use Matlab as the development environment and CML as the simulation software, we considered several other alternatives. There are several commercial packages that are optimized for physical-layer simulation of communication systems, including VisSim/Comm[6], SystemView, and LabView. These alternatives require software licenses and do not make the source code available for review. While some of these products support the same types of modulation and coding as CML[7], none support the Monte Carlo evaluation of constrained channel capacity discussed in Section 3.2. In addition to these physical-layer simulation tools, there are commercially available network simulators, such as OpNet. While OpNet could possibly be used to support the ad hoc networking research discussed in Section 3.3, it suffers from expensive licensing requirements and does not support physical-layer simulations required for the research projects discussed in Sections 3.1 and 3.2.

Matlab markets a *communications toolbox*, which can be used to write basic simulations in script format. It has support for linear modulation such as PSK and QAM and convolutional and basic block coding (e.g. Hamming, BCH, and Reed Solomon). However, it does not support nonlinear modulation (e.g. FSK), capacity approaching codes (e.g. turbo, LDPC), or capacity calculations. Matlab also markets a *communications blockset*, which allows communication systems to be implemented within *Simulink*. With this alternative, simulations are laid out graphically by interconnecting blocks in a diagram, in contrast with writing a script. The communications blockset has more features than the communications toolbox, for instance support for FSK modulation and soft-output decoding of convolutional codes. However, it does not have support for LDPC codes or capacity calculations. Besides not having full functionality, the communications toolbox and blockset must be licensed and are not available in the student version of matlab. In addition, the source code that underlies the functions and blocks is generally not available for inspection by the user.

In addition to the above commercial options, there are free software options available, including IT++, Ptolemy, and ns2. While each of these projects can meet some of our design objectives, none can meet all of them. We have essentially already developed the majority of the simulation software that is needed to successfully complete the project, and it would be inefficient to restart with an entirely new and unknown software product.

---

[5] We have confirmed through collaboration with researchers at Motorola that CML works in Octave, although with a significantly slower execution speed than when run in matlab.

[6] http://www.vissim.com/products/comm/index.htm

[7] For instance, the turbo code toolbox in VisSim/Comm was developed by the PI and is derived from an earlier version of the Coded Modulation Library.

4.2 Grid Computing Architecture

The goal of the grid computing architecture is to support the required computing infrastructure with minimal hardware investment. The key to accomplishing this goal is to marshal idle capacity on existing general-purpose computers located initially at the proposing institution, and later at the institutions of other users of the resource. The machines that comprise the grid may be located in libraries, student labs, or the desks of faculty, students, or staff. The machines could run a wide assortment of operating systems, including different versions of linux and windows. Each machine would be required to download a small and lightweight software application called the *compute engine*. It is the compute engine that turns the PC into a node on the grid.

When a researcher has a job to start, he or she selects the desired simulation parameters on the web-portal and clicks a button to start execution (while the web-portal is under development, simulations will be launched to the grid directly from Matlab). The information entered on the web page is processed into a data file that specifies the requested simulation parameters. The code containing the simulation logic has been implemented using a combination of Matlab and C, and by using the Matlab Compiler, it can be compiled into a stand-alone executable. During the initial development phases, the compiled executable file (which is about 200 Kbytes) and the simulation parameter file (about 7 Kbytes) is sent over the Internet from the server to an available grid node (i.e. a machine that is not otherwise being used by a local user or other grid job), where it will be executed.

In order to run compiled Matlab code, the grid node must have installed the *Matlab Component Runtime (MCR)*. While this library is available for free from the Mathworks, it takes up about 300 Megabytes of disk space and care must be taken to assure that it is the correct version. We do not foresee this as a barrier to running jobs on our own grid nodes, since we can control what is loaded onto those machines, but as participants from other institutions begin to contribute their own nodes to the grid, we would like to get away from using compiled code and requiring the *Matlab Component Runtime*.

The solution we propose to obviate the need for the MCR is to translate the simulation code into Java. The benefits of using Java extend far beyond not needing to preload each machine with MCR. Java is platform independent, hence simulations can run on any machine, regardless of its operating system (using compiled code, different executables must be generated for each operating system). Furthermore, using Java can assure a certain level of security. Native code (e.g., C/C++ or Fortran) generally has full access to the system, and therefore can potentially be harmful, either from the malicious infection of the simulation library or from an accidental software bug (e.g. a memory leak). Because Java code runs within the safe confines of the Java Virtual Machine, it can be isolated from accessing restricted areas of the system. For instance, it can be prevented from accessing Internet ports, memory outside its heap or the file system

Once launched, the job will run on a compute engine. In earlier stages of development, there will be a one-to-one mapping of jobs to compute engines, but later in the development cycle we will develop methodologies for parallelizing individual jobs so each can run on multiple compute engines. At periodic report intervals (5 minutes), the state of the simulation is saved to a file and the file is sent back to the server. The latest state file can be retrieved by the user from the web portal, hence the user will always be able to access simulation results that are no more than 5 minutes old. If a task is interrupted on a particular compute engine, it is automatically rescheduled by the server to another one.

After surveying the alternatives, we have decided to contract grid service through the *Global Grid Exchange (G2EX)*[8], an initiative of the *West Virginia High Tech Consortium (WVHTC)*. The Global Grid Exchange is based upon the *Frontier* software platform, developed by *Parabon Computation*[9]. A server, which is hosted by WVHTC, receives job requests from the user base, and then schedules jobs to nodes on the grid. Results are returned back to the server, and the user can query the server for results when desired. In this "launch-and-listen" paradigm, the user can start a job from a notebook computer, shut down the notebook, travel across the country, boot up the notebook, and retrieve the results.

The Global Grid Exchange can schedule jobs either to a large public grid, the nodes of which it has recruited from various sources, or, in our case, a virtual private grid (VPG) the nodes of which are designated by participating institutions for inclusion in this initiative. For security reasons, G2EX disallows the execution native tasks on its public grid. On a VPG, however, where participating institutions contribute their own nodes, G2EX permits native task execution by authorized users. Under this paradigm, the cost is a monthly fee per compute engine that can be packaged as a "software license". The other option is to run tasks on a grid of about 3,000 machines that the Global Grid Exchange has recruited and manages. In this paradigm, costs are assessed on a per-CPU hour basis, but tasks must be written in Java (native code is not generally allowed). The overall costs are highly competitive, reducing greatly the costs normally associated with high performance computing (see Budget Justification portion of this proposal for further discussion).

To confirm that the Global Grid Exchange will meet the needs of the proposed infrastructure, we have conducted a proof-of-concept test. Certain portions of the CML library, included those necessary to support the research projects described in sections 3.1 and 3.2, were launched from matlab and executed on a VPG comprised of 30 linux machines in one of the teaching laboratories at WVU. This trial was actually used to generate the simulation results shown in Fig. 2.

It is noted that Sun Microsystems offers a similar pay-per-use grid service [10]. However, the Sun grid service runs on its own dedicated compute infrastructure and does not follow our paradigm of leveraging idle resources at the participating institutions. As a consequence, the costs of using Sun's service are high ($1 per hour), while our cost of using G3EX is only 10 cents per hour.

An alternative to contracting grid service would be to construct and manage a grid on our own. There is a growing set of free and proprietary software and projects that could be used for this purpose, for instance the Globus Toolkit[11], Condor[12], TeraGrid[13], the GridPort Toolkit[14], GridMP from United Devices[15], and many others. However, since the expertise of the proposing team is primarily on the scientific principles behind communication systems, and not specifically on grid computing technology, it is more practical to use the proposed solution, especially since we have already confirmed that it works and can be used at relatively low cost.

---

[8] http://www.globalgridexchange.com/
[9] http://www.parabon.com/
[10] http://www.sun.com/service/sungrid/index.jsp
[11] http://www.globus.org/
[12] http://www.cs.wisc.edu/condor/
[13] http://www.teragrid.org/
[14] http://gridport.net/main/
[15] http://www.uniteddevices.com/

5. Implementation Plan

This section discusses how the infrastructure will be developed, using a 4 phased approach.

5.1 Phase I, A VPG Accessed via a Matlab-Based Interface

We will begin with a relatively small virtual private grid (VPG) with nodes hosted at the proposing institution. It will use computers in teaching classrooms/labs plus a 30-node cluster that we are requesting in the budget, which will provide a baseline level of service plus a platform for testing. Altogether, there will be about 100 nodes in the VPG at the onset of the project.

Because the VPG nodes will be under the control of participating institutions, they can be configured to run native code, each preloaded with the Matlab Component Runtime. Grid jobs will be launched from matlab by first preparing a data file that specifies the simulation parameters and derived values (code rates, signal constellation diagrams, interleaver designs, etc). This data file will be sent along with the compiled matlab code to a grid computer, where it will run until completed. This methodology was used in our proof-of-concept test, and so we have confirmed that it works. We will extend this work to incorporate all types of simulations supported by CML, including simulation of ad hoc networks described in Section 3.3, and will allow execution on windows machines as well as linux machines.

The initial implementation requires that grid jobs be launched and managed from within Matlab. A set of tools will be developed that will allow the user to monitor the progress of their simulation from within Matlab by typing in commands at the Matlab prompt.

5.2 Phase II, Web Portal and Opening the Grid to the Community

Although Matlab is prolific within the research community, a disadvantage of requiring Matlab is that not all organizations will have access to it, and the text-based interface to the proof-of-concept grid service environment has a rather steep learning curve. To make the project accessible to a wider audience, we will develop a web portal during the first year of work from which users can more easily configure and manage jobs.

The web portal will allow the user to set up simulations, monitor their progress, and download data files. While WVU will develop portions associated with setting up the simulations, we have decided to contract some of the web development effort to Parabon Computation, the company that developed Frontier, the grid platform (including compute engine) that powers the Global Grid Exchange. More specifically, Parabon will develop the portion of the web-portal that monitors grid jobs and user accounts (e.g. how many hours remain); given its knowledge of the grid architecture, it will be most cost effective to have Parabon perform this part of the development.

To accompany the web portal, we will develop a graphical user interface (GUI) for viewing and plotting results. The GUI application can be developed in matlab, but packaged as a stand-alone executable so that the users will not need a matlab license to view their results.

Once the web-interface is completed, the project will be opened to the research community. Organizations can apply for accounts and an initial set of CPU hours. From the web-interface, they will launch simulations, but the simulations will still run on the VPG maintained by the proposing institution. We anticipate that this milestone will occur at the end of the second year, with the first partner organizations performing beta-testing of the infrastructure.

5.3 Phase III, Port to Java and Inclusion of Remote Nodes

Once the web portal is made accessible to the research community, we anticipate a growing demand for computing power. Some of this demand will be met through the inclusion of more nodes contributed by WVU. We anticipate that in year 2, the number of nodes supplied by the proposing institution will grow to between 200-300 (WVU Libraries alone could easily contribute 200 nodes). While WVU could continue to recruit more and more internal nodes, the long-term sustainability of the project requires that outside research organizations that use the resource should be encouraged to contribute nodes on their own.

However, there are some technical obstacles associated with remote nodes that are outside our immediate control. To run compiled matlab code, these machines must be loaded with the appropriate version of Matlab Component Runtime, and the compute engines configured to allow native codes. Outside users might be uncomfortable with the prospect of running unknown native code on their machines, and the associated security implications. Also, upgrading to new versions of Matlab will become logistically challenging, as all nodes on the grid must upgrade their version of MCR. Finally, the compiled matlab paradigm requires a copy of the Matlab compiler for each type of architecture (linux, windows, Mac OSX) which adds to the cost and inconvenience.

To alleviate these problems, we propose porting to Java the (remotely executed) task portion of the simulation software. Frontier's security model allows safe and secure execution of Java tasks eliminating the risks to which providers of compute nodes would otherwise be exposed. A survey of the current release of CML reveals that the amount of code that must be translated is about 2,000 lines of matlab code and 4,000 lines of C code. We anticipate that once translated, the amount of Java code will be approximately the same, i.e. 6,000 lines. We estimate this translation project can be accomplished in one year by a group of 4 undergraduate and graduate students, and completed by the end of year two. Note that the current release of Matlab includes a product called "Matlab Builder for Java" which allows for the automated translation of matlab code to java classes [16]. We have requested funds to purchase a license for this product, and anticipate that it will help ease the transition. However, we believe that simulation code that is repetitively executed should be optimized by hand.

Once the Java translation task has been completed, we will allow organizations that use the resource to begin contributing their own computers to the grid. We anticipate that this will increase the number of nodes during year three to between 300-500 nodes and during year four to 500-1,000 nodes. In order to achieve this target, we require most organizations to exchange CPU hours at a 1-for-2 rate, i.e., an organization would be expected to contribute 2 hours of CPU time for every hour of runtime consumed by their own jobs. To broaden the impact, we will make an exception for certain institutions (whose researchers initially get 4,000 hours of CPU time), which will be able to exchange CPU hours on a one-to-one rate.

Another benefit of converting the project to Java is that it opens up a larger set of computing nodes. At the time that the proposal was written, the Global Grid Exchange has recruited about 3,000 computers on which it has authorization to run Java code. These nodes could be used on an as-needed basis and can be used to help balance supply and demand.

---

[16] http://www.mathworks.com/products/javabuilder/

5.4 Phase IV, Increase Cluster Size and Parallelize Individual Jobs

During year three, organizations will begin contributing their own nodes as demand for services increases. However, the baseline CML library is not expected to meet the needs of all the users, especially those working on the forefront of the field. As new concepts are introduced to the research community, for instance new types of channel codes, modulation formats, or ad hoc networking protocols, there will be a demand for users to run their own code on the grid. In the final two years of the project, we will develop a plug-in methodology through which users can incorporate their own code into system. The code would need to be written in Java and would be called from other portions of the CML library. While users would be free to run their code privately on the grid, we would encourage users to contribute it to the library so that it is available for use by other researchers. As the project gains momentum and contribution from the research community, we would act as software integrators, tying in contributions into the mainline release. Although this project is conceived as a tool to support the communications and information theory research communities, once user supplied Java code is supported, we anticipate that the project will begin to appeal to a much wider scientific computing audience.

During the initial stages of development, we plan to use a one-to-one mapping of jobs to grid nodes. A single simulation would run on a single node until completion. However, this ignores the fact that simulation of communication system is inherently parallelizable. For instance, a typical simulation of a turbo coded system might run until 30 independent codeword errors are counted for each value of signal-to-noise ratio. While this simulation could run on a single processor, it could be distributed to 30 machines, each with an independent random seed, and run until each machine encounters 1 error per SNR point. Data would need to be aggregated at the server, which would create a single coherent simulation state file that could be downloaded by the user. In the final year of funding, we will introduce support for parallelization into the system. When a job is requested, the server will send out multiple identical jobs that will run on several machines, each with its own seed. After a certain number of Monte Carlo trials (e.g.. 1,000), each machine will report back the state of its own local simulation. The server will combine into a global simulation state file, and will halt once the necessary number of errors has been logged.

5.5 Plan for Long-Term Self-Sufficiency

While the budget is sufficient to sustain a level of 500-1000 nodes until completion of NSF funding, sustainability of the project after NSF involvement will require that some organizations contribute not only nodes and software, but also financial support. Fortunately, once developed, the system can exist with a minimal amount of funding (about $60K/year to keep a 1000 node cluster running), but this funding should come from the user base itself. To generate the necessary level of support, we will open the grid to commercial use. Companies could contribute financial support by purchasing CPU hours (we anticipate a rate of about 20 cents per CPU hour). If the demand for CPU hours exceeds the computing power supplied by the user base, then we could purchase additional CPU hours from the Global Grid Exchange (at a rate of about 10 cents per CPU hour). Note that this rate is much less than the $1/hour charged by Sun Microsystems.

6. Educational Plan

The proposed infrastructure can be used for more than just research. It also has an educational impact by providing a hands-on experience to students, allowing them to test their designs and quickly get results back. We plan to feature the proposed infrastructure in several courses by integrating it into both the lecture and the outside assignments. During certain targeted lectures, the instructor would log into the webpage and show students how to use the tool and help them interpret the results that it produces. On-line tutorials can be developed that asynchronously lead students through complex topics, allowing them to experiment with the concepts via simulation. Classes would include projects that require students to engage in design work and then test their design via simulation.

6.1 Enhancement of Specific Existing Courses

We envision three specific classes that would benefit from the proposed infrastructure, namely *Coding Theory*, *Information Theory*, and *Wireless Networking*. All three of these courses are currently being taught at WVU. The benefit to WVU is that the learning experience in these courses will be reinforced through the assignment of projects that require the use of the proposed infrastructure.

6.1.1 Coding Theory

Coding Theory is concerned with the mathematics of forward error correction coding. It requires knowledge of discrete mathematics, matrix-vector theory, and probability, assets that are important to both the modern Electrical Engineer and Computer Scientist. The course involves a certain amount of design, namely the design of specific error correction codes. In the course, which at WVU is taught at the graduate level (as EE 567), students will be asked to develop their own LDPC codes and test their design using the proposed infrastructure. The code will be specified in a file that describes the parity check matrix which is uploaded by the student to the server. After the simulation has concluded, the student can retrieve results form the server. Running such simulations in parallel over the grid will allow more immediate feedback to the student.

6.1.2 Information Theory

Information theory is the mathematical foundation of information technology in general, and communications technology in particular. It deals primarily with fundamental limits of communication and computation, and is something that the modern Computer Scientist and Electrical Engineer should be aware of. In the course, which at WVU is also taught at the graduate level (as EE 568), students will be able to use the proposed infrastructure to determine the capacity under modulation constraints. Students would prepare a file that describes a certain type of modulation, upload the file, and then see their results.

6.1.3 Wireless Networking

As wireless networks have become pervasive, there is a growing need to educate Computer Scientists and Engineers that understand how they operate. West Virginia University has recently developed a senior level undergraduate course on wireless networking (CPE 462) that is targeted at computer engineers but is also appropriate for computer scientists. Unlike the other two courses (Coding Theory and Information Theory) the course will be offered at the undergraduate level. In the course, a project will require students to lay out a wireless network and test its performance using certain protocols, and the project will require the use of the proposed infrastructure.

7. Dissemination Plan

The proposed infrastructure is based on the CML toolbox, which is licensed as open source software under the lesser GPL. We will continue to make the software available, and users will always have the option to run it on their own local computer. We will also advertise the ability to run CML-based simulations on the grid infrastructure by describing this ability on the CML webpage and giving presentations on this capability whenever possible (for instance, the conference paper [ToCh07] mentions how the grid computer was used to generate results).

To give specific training to the user-base, we propose giving free tutorials on using CML at one conference per year, such as ICC, WCNC, and ISIT. The WCNC conference in particular is a good target because WCNC has a new policy of offering one free tutorial to each registrant and therefore has a need for tutorials. The ICC 2009 conference is also a good candidate because the PI has a prominent role in its organization (co-chair for the Wireless Communication Symposium). For an example presentation given by the PI on how to use CML, see: http://www.csee.wvu.edu/~mvalenti/documents/CmlTheory.pdf

8. Project Timeline

A summary of project milestones are described below.

Year One (Apr. 2008 to Mar. 2009):

- Quarter I: Project kickoff: Recruit initial student assistants.
  First tutorial on using CML (WCNC 2008).
- Quarter II: Run compiled Matlab simulations on virtual private grid at WVU.
- Quarter III: Scale grid to 100 nodes using nodes from the CSEE department.
  Develop tools to monitor simulation progress.
- Quarter IV: Develop projects for Coding Theory course.

Year Two (Apr. 2009 to Mar. 2010):

- Quarter I: Second tutorial (ICC 2009).
- Quarter II: Develop prototype of web-portal.
- Quarter III: Complete and test web-portal design.
  Recruit additional nodes at WVU, bringing total to 200-300.
- Quarter IV: Develop projects for Information Theory course.
  Allow outside "customers" to run simulations from the web-portal.

Year Three (Apr. 2010 to Mar. 2011):

- Quarter I: Third tutorial (ISIT 2010).
  Begin porting CML code to Java (using *Matlab Builder for Java*).
- Quarter II: Develop projects for Wireless Networking course
- Quarter III: Complete port of CML simulation code to Java (hand-optimization).
- Quarter IV: Recruit nodes outside WVU, bringing total to 300-500.

Year Four (Apr. 2011 to Mar. 2012):

- Quarter I: Fourth tutorial (ICC 2011)
- Quarter II: Parallelize the simulation code
- Quarter III: Institute policies for self-sustainability: Customers of the resource must contribute grid resources or else pay for service.
- Quarter IV: Recruit nodes from partner organizations, bringing total to 500-1000.
  Complete documentation.

9. Summary of the Broader Impacts

As described in the recent National Research Council report *Renewing U.S. Telecommunications Research* [NRC07], there has been a decline in support for fundamental research in telecommunications. This decline places the leadership position of the U.S. in the telecommunications industry at risk. To maintain its strength, the U.S. must place an emphasis on high-value innovations, since it cannot compete with commodity services and products that can be outsourced to countries where the costs are lower. However, high-value innovation in telecommunication research requires high performance computing resources to simulate and test new technologies. Because only the largest and most well-known institutions receive enough federal assistance to create their own high performance resources, a need exists for researchers at other institutions to run high-performance simulations of telecommunication systems.

The main broader impact of the proposed infrastructure is that it provides the ability to run massive simulations of communication systems to any researcher that has Internet access. By opening up the infrastructure to the telecommunications community, the digital divide that separates large research organizations from the rest of the research community can be broken down. By enabling a larger diversity of researchers and organizations to participate in advanced telecommunications research, the position of the U.S. as leader in telecommunications technology will be less likely to slip away.

The infrastructure also has educational impacts, as it can be used as the basis for student projects in courses on coding theory, information theory, and wireless networking.

References Cited

[3G02] Third Generation Partnership Project 2 (3GPP2), "Physical layer standard for cdma2000 spread spectrum systems, release C," *3GPP2 C.S0002-C Version 1.0*, pp. 115–122, May 28 2002.

[BeGl93] C. Berrou, A. Glavieux, and P. Thitimasjshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes(1)," in *Proc., IEEE Int. Conf. on Commun.*, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[CaTa98] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory*, vol. 44, pp. 927–946, May 1998.

[CaTu01] G. Caire and D. Tuninetti, "The throughput of hybrid-ARQ protocols for the Gaussian collision channel," *IEEE Trans. Inf. Theory*, vol. 47, pp. 1971–1988, Jul. 2001.

[Ed06] C. Edwards, "Intel's $600 million WiMax bet," *Business Week*, July 6, 2006.

[ETSI00] European Telecommunications Standards Institute, "Digital video broadcasting (DVB); interaction channel for satellite distribution systems," *ETSI EN 301 790 V1.2.2 (2000-12)*, 2000.

[ETSI03] European Telecommunications Standards Institute, "Digital video broadcasting (DVB) second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications," *DRAFT EN 302 307 DVBS2-74r15*, 2003.

[ETSI05] European Telecommunications Standards Institute, "Universal mobile telecommunications system (UMTS): Multiplexing and channel coding (FDD)," *3GPP TS 125.212 version 6.6.0*, Sept. 2005.

[Ga60] R. G. Gallager. *Low-Density Parity-Check Codes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1960.

[Ga68] R. G. Gallager, *Information Theory and Reliable Communication*. New York: John Wiley & Sons, Inc., 1968.

[Go05] A. Goldsmith, *Wireless Communications*, Cambridge University Press, 2005.

[Ha50] R. W. Hamming, "Error detecting and correcting codes," *Bell Sys. Tech.J.*, vol. 29, pp. 147–160, 1950.

[IEEE05] Institute for Electrical and Electronics Engineers, *IEEE Std 802.16e-2005*, Feb. 2006.

[Ka04] K. Seki, T. Itabashi, T. Higuchi, Y. Kasai, and E. Takahashi, "Performance evaluation of low latency LDPC code," *P802.3an Task Force Meeting*, (Portland, OR), July 2004.

[MaCo98] S. J. MacMullan and O. M. Collins, "The capacity of orthogonal and bi-orthogonal codes on the Gaussian channel," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1217–1232, May 1998.

[NRC01] National Research Council, *Embedded, Everywhere*, National Academy Press, 2001.

[NRC02] National Research Council, *Broadband: Bringing Home the Bits*, National Academy Press, 2002.

[NRC06] National Research Council, *Renewing U.S. Telecommunications Research*, National Academy Press, 2006.

[Pe01] C. E. Perkins, *Ad Hoc Networking*, Addison Wesley, 2001.

[Sh48] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379–423 and 623–656, 1948.

[ToCh06] D. Torrieri, S. Cheng, and M.C. Valenti, "Robust frequency hopping for interference and fading channels," submitted to *IEEE Trans. Commun.*, Oct. 15, 2006 and May 25, 2007.

[ToCh07] D. Torrieri, S. Cheng, and M.C. Valenti, "Robust frequency-hopping system for channels with interference and frequency-selective fading," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, (Glasgow, Scotland), June 2007.

[VaCh05] M.C. Valenti, S. Cheng, and R. Iyer Seshadri, "Turbo and LDPC codes for digital video broadcasting," Chapter 12 of *Turbo Code Applications: A Journey from a Paper to Realization*, Springer, 2005.

[ZhVa05] B. Zhao and M.C. Valenti, "Practical relay networks: A generalization of hybrid-ARQ," *IEEE J. Select. Areas Commun.*, vol. 23, no. 1, pp. 7-18, Jan. 2005.