CS 514 Course Project

This handout is intended to get you started on your CS 514 course project. We've organized it into three parts. The first part concerns our goals for the project. Whether you do one of the topics we came up with or invent one of your own, the key requirement is that your project must achieve these goals. The second part describes two of our ideas for possible projects. You are *not* required to use our ideas or even the software tools we suggest that you consider working with. Ultimately, that is, choice of a project is up to you. The last part of this handout describes what you need to hand in, and when.

One thing we mentioned on day one (and on the home page) but that should be stressed again is that *no extensions will be permitted on these projects.* The basic due date will be the last day of classes – we're thinking of doing a sort of project show-and-tell that day in the CS UG Lab on the third floor of Upson. CS514 has a posted final date, and absolutely no work can be handed in beyond that date. We'll base your grade (project and all) on what we have as of the afternoon of the final day. Projects handed in late (after the show-and-tell day but by the date of the final) will be marked down by five points per week (out of 100 points maximum), so that handing a project in after the show-and-tell date won't give the team doing the project an advantage.

Part I: Our goals

The purpose of your project is to gain hands-on experience with state of the art middleware tools (such as C# for .NET, Web Services, ASP.NET, etc – or Linux, CORBA, C++ if you prefer that sort of platform) and to use them in a setting similar to the kinds of things many people will be doing in the coming decade or so in industry. In addition, the project is a chance to learn practical skills from online learning resources and demos, and also to learn to work in a small team if you opt to do so (teams of two or three are encouraged but certainly not required). The CSUG lab is available to you for your work, but you can work on other platforms using other software as long as we can see a demo without taking a long hike to your dorm room on the show-and-tell day.

The project needs to reflect the themes of this course. Up to now, those themes have been somewhat high level – gaining familiarity with the major components of a technology like Web Services or the major elements of an Internet application. But the remainder of the course will dive down into much more detail in some specific areas where we see a lot of value. For example, you'll learn much more about fault-tolerance and replicating data or services than you ever wanted to know! We expect that projects will draw on this knowledge in interesting ways – not blindly, but that you'll think hard about how to use some of these ideas to overcome specific issues that arise in real systems. If you can use these same tricks in a job two years from now, CS 514 will have been a big success in our eyes.

Part II: Some suggestions for possible projects

In fact, we are pretty flexible about what we'll access – look at some of the older course web sites for additional ideas. But here are two fresh new ideas:

1. **Adding Better Fault Tolerance Support for Web Services.** As you know, if a client has a timeout or other problem in a Web Services setting, failing over to a backup can be a challenge. It isn't obvious how to find that backup (in a naming sense), how to force Web Services to rebind to it, or how to resynchronize so that the user won't see a perceptible disruption – e.g. you don't want an air traffic system to forget about a command that was supposedly accepted by the system before the fail-over, or to behave inconsistently. Our first idea is to have you build a solution to these problems and to evaluate the performance and scalability of the resulting system.

   Here's what we want you to do. This may sound "high level" but we don't plan to give more detail. Working out the details will be part of your job.

   Using a Web Services technology – we suggest the ASP.NET template from Visual C# .NET – build a simple client-server application that can somehow demonstrate basic features of a Web Services architecture. You might want to build a fake air traffic control system to make it look more realistic. The server in that case would keep track of airplane flight plans and trajectories. You don't need to have a real database behind the server, but if you feel like doing so, ASP.NET includes a simple way to use MS SQL Server for this purpose. If not, your Web Server program should have a data structure of its own.

   Measure the performance and scalability of your solution. How fast is it, and how is it impacted when more and more clients connect to the server? Can you estimate the maximum load it could achieve? Where do the bottlenecks seem to be? Could caching or other tricks speed it up?

   Next, we would like you to implement a fault-tolerance mechanism. On the server side, you'll want to change your server to run in a group configuration with 2 or more servers in a group (ideally the group size could be varied depending on load). Think about the pros and cons of load-balancing clients over this group. How could that be done? Which approach would work best?

   Now, we want you to extend the server so that the state it maintains is replicated among its instances – e.g. the MS SQL server data has to be visible to many servers or the actual data managed needs to be replicated. If you make MS SQL server data available to many Web Servers is it better to have multiple instances of MS SQL server, or to have one instance that several front-ends can access? Keep in mind that a system like an air traffic control console needs 24x365 continuous uptime. Ten minute pauses to reboot SQL Server would be extremely dangerous!

You could implement your own data replication mechanism, but if so it needs to be fault-tolerant. Alternatively, you can download the Cornell Ensemble package or some other package such as JavaGroups from the web. These will do data replication and fault handling for you. There are many such packages available, including the ones mentioned, Eternal (a package for CORBA), Spread (one for C applications developed at John Hopkins University) and others. If you go this route, factor in two weeks or so to get the hang of using the package you decide to employ and for figuring out how to integrate your Web Services system with the package! If you build your own solution, of course, that will take a few weeks too. So don't delay this stage of your project or you are certain to end up with a lame result and a low grade.

Finally, implement a client "fail-over" mechanism that resynchronizes with the servers and reissues any pending operation if one was underway at the time of a failure. We do not require that you support fail-over in the middle of a Web Services transaction but if you can pull that off, we'll be very impressed.

That's it for implementation. Beyond this we want you to evaluate the scalability and performance and bottlenecks of the modified system and prepare a simple poster-style presentation for the demo day explaining what you did, how it worked before the replication mechanism was added and how it worked afterwards. During the demo, you should try and impress us with the scalability, speed and quick failover capabilities of your solution. All group members should be able to answer questions about any aspect of the solution. All should have written comparable amounts of code, and comparable difficulty. You'll also be handing in a very short project summary that we can reread offline.

People doing this as an MEng project are expected to do more than a minimal solution. A person working by themselves and just for CS514 might get maximum credit for a considerably less ambitious "bare bones" solution.

2. **Peer-to-Peer Application through NAT boxes.** Peer-to-peer (P2P) applications are gaining in popularity, in part because they offload servers from the "heavy lifting" of high volume exchanges—file transfers or video streams. Often P2P applications fail because they cannot deal with NAT boxes or firewalls correctly. For instance, the Yahoo Instant Messenger (IM) application recently added the capability to send high-resolution webcam images directly between peers. In Paul's experience, however, this fails to operate through NAT even though there may be considerable bandwidth between the peers. Our second idea is to have you build a P2P application that operates robustly through NATs.

The core idea is for the peers to use a "rendezvous" server or servers of some sort to discover each other's presence, and to determine what port and address to use to make direct contact between peers. The preferred approach for this will be for you to use STUN and dynamic update of DNS SRV records. However, you may substitute protocols of your own design if you wish. Either way, the system must be robust to server failure. (Note that DNS has fault tolerance built in, so using this component will give you a leg up.)

The application should have a presence component (a way for the peers to learn of each other's existence and readiness to participate), and some simple application such as chat, file transfer, or a game. A two-party application is certainly sufficient, but if you really want do an N-party application and don't care to have a life outside of CS514, then go for it.

In addition to writing the application, you'll have to design a test environment and demonstrate that your application works in a variety of cases (both peers behind different NAT boxes, both peers behind the same NAT box, one or both peers behind two NAT boxes, one or both peers not behind NAT boxes). Note that the test environment may require that the lab managers install new software or boxes, so we will work with them on that as needed.

At your demo, you must show that the discovery phase of your application works even when one of the (redundant) servers becomes unavailable. You must also show that the P2P part of your application, including the presence aspect, works even when ALL servers have become unavailable. You must be prepared to explain how the use of servers has been minimized for maximal scalability.

For your application, you may use any set of software tools you wish. You may even use existing applications if you can prove that the application does not itself work P2P through NAT boxes.

Note that this project is at the state of the art (even Yahoo has not managed to get this right!), so you can expect glitches along the way. Open source STUN implementations are available, but we are not sure how mature they are. To our knowledge, nobody has used dynamic update of DNS SRV records as a way to do P2P rendezvous, so again there may be unexpected problems. (If you use DNS updates, then the tool will be BIND 9 from www.isc.org. If STUN, we're not yet sure all the options.) We expect you to identify these glitches along the way, and find ways around them.

Part III: Handing stuff in.

You'll be handing things in at the end of each of five stages. One week from now we expect you to hand in a project plan describing the project you have decided to tackle and laying out a timeline for doing it. Where will you be after three weeks? Six weeks? After nine weeks?

In the first written report, people who decide to do a "non-standard" project need to explain exactly what they plan to do and hope to accomplish and how it relates to course themes. If you can do this without much of a stretch, we will definitely approve your proposal.

At these three week intervals we would like a very short status report. CS514 homework will diminish later in the class and we expect you to be hard at work on your project in the second half of the semester. If you slip on your schedule and need to reduce functionality, your status report should summarize the issues and what you plan to do.

On the last day of classes we'll hold our show-and-tell. For that, bring a poster (Kinko's can help you make it, or just print things and paste them on a suitable poster board) showing the architecture of your overall system (one of those pictures with lots of boxes and arrows), a brief high level summary of the capabilities you focused on and the tools used. For the fault tolerant web services project, show performance before and after the replication and fault-tolerance mechanism was introduced. A fancier report might drill down to identify critical paths, show scalability in an elegant way, etc. Please print (black and white) the same material with the names of all group members clearly shown on page one and hand this in, with an additional one or two pages of written material covering what you will be presenting orally during the show-and-tell. This should summarize what you set out to do, what you actually did, and how well things worked out.

HELP! Who will teach me C#? How can I learn about STUN? Where are the documentation books?

C# first. As noted early in the course, this is an advanced course for people with a great deal of background from courses like CS314 and CS414. We assume that you are a strong programmer in C, C++, Java or some similar language. If so you can probably do Paul's project without learning anything radically new. The Visual C# Web Services project will require learning new stuff, though.

Since C# and Java are so similar, if you know Java this transition will be easy. If you are a C or C++ programmer the issues are perhaps a bit more subtle. If you don't know any sort of object-oriented language, however, it may be hard to learn C# fast enough to become reasonably productive on .NET so start early or work in some other language that you know better. A project can be done in LISP, Fortran, Python, Tcl/Tk, Visual Basic, O'CaML or anything you like. In fact, many of these are even supported in .NET!

Cornell has many online resources for learning C# built right into Visual C# .NET. You can also buy printed materials in the Cornell bookstore, but you probably can do everything without opening a single book if you stick to our suggested projects.

For a person with reasonable background these will be adequate to jump in and, after a few bumps and groans, you should be able to do anything. Start Visual C# for .NET, tell it to create a new project, select the "windows forms" template and begin playing with it – eventually, this can become your client application. We suggest that you begin with an application that displays "Hello World" in a pop-up dialog box that the user can dismiss by clicking OK. (Hint: This specific Hello World application is actually included in the help materials and isn't hard to find). As you get comfortable you may want to think about having multiple forms (multiple windows), threads, etc.

To build the server you'll use the ASP.NET template. ASP.NET has a tutorial for building a simple Web Service. Follow the instructions for doing one of the demo projects and then build your way "out" by modifying it into what you need for the course.

People wishing to do Paul's suggested project can work without quite so much infrastructure. The tools he has in mind lack the same sort of extremely detailed online tutorials you'll find in the help pages of Visual C# .NET (which has the most comprehensive online help system any of us have ever seen). STUN in particular will be bare bones. But if you come from a very remote background and don't have the time or ability to teach yourself C#, that project may be easier. Again, no matter what project you pick, we are ok with you using a language you already know or packages downloaded from the Web.

If you are completely unable to write code and unable to teach yourself some sort of language, we aren't going to be able to help very much. It just isn't the role of CS514 to teach introductory programming, even in this new language and new system. ***On the other hand, if you do dive in and then get stuck, go to the class news group first, and if that doesn't work go to Rimon***. If necessary, he'll refer you to one of us as a last resort.