Universidade Federal de Campina Grande

Centro de Ciência e Tecnologia

Coordenação de Pós-Graduação em Informática

Dissertação de Mestrado

Reputação Autônoma como Incentivo à Colaboração no Compartilhamento de Recursos Computacionais

Nazareno Ferreira de Andrade

Campina Grande, Paraíba, Brasil

Março - 2004

Reputação Autônoma como Incentivo à Colaboração no Compartilhamento de Recursos Computacionais

Nazareno Ferreira de Andrade

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Walfredo Cirne

(Orientador)

Campina Grande, Paraíba, Brasil

Resumo

A computação em grid surgiu como uma forma de agregar uma quantidade arbitrária de recursos pertencentes a diferentes domínios administrativos. Contudo, compor um grid computacional hoje envolve a negociação humana com os responsáveis pelos recursos. A necessidade dessa negociação limita a escala dos grids usados em produção hoje. Existem propostas para resolver esse problema criando uma economia baseadas em moeda, porém nenhuma dessas propostas é amplamente usada em produção. Como alternativa a essas propostas, foram propostos sistemas de compartilhamento de recursos par-a-par. Essa abordagem, embora não resolva o problema geral da composição de grids, permite ter soluções de fácil implantação e adoção a curto prazo. O OurGrid é um grid par-a-par focado em aplicações Bag-of-Tasks (aplicações paralelas cujas tarefas são independentes) que está sendo desenvolvido na UFCG em colaboração com a HP. Como na maioria dos sistemas de compartilhamento par-a-par, existe no OurGrid uma preocupação com os usuários que não colaboram com seus recursos para os sistemas, apenas consumindo os recursos compartilhados. Esta dissertação apresenta a Rede de Favores, um esquema de reputação par-a-par autônomo que objetiva incentivar a colaboração no OurGrid. Através do comportamento autônomo dos pares do sistema, os pares que colaboram mais são priorizados quando requisitam recursos da comunidade. Sendo inteiramente autônoma, a Rede de Favores tem fácil implementação e implantação, permitindo seu uso em produção no OurGrid sem adicionar complexidade signifi cativa à sua adoção.

Abstract

Grid computing emerged as a way of gathering an arbitrary amount of resources from different administrative domains. However, assembling a computational grid nowadays involves human negotiation with the resource owners. The need of this negotiation limits the size of the grids used in production today. There are proposals for solving this problem creating a currency-based economy, but there is no currency-based solution largely used in production currently. As an alternative to these proposals, some have proposed peer-to-peer resource sharing systems. Although this proposal doesn't solve the general grid assembling problem, it allows building solutions of easy deployment and adoption in short term. OurGrid is a peer-to-peer grid that targets Bag-of-Tasks applications (those parallel applications whose tasks are independent) being developed in UFCG in collaboration with HP. As in most peer-to-peer systems, free-riding is an issue that must be addressed in OurGrid. This dissertation presents the Network of Favors, an autonomous peer-to-peer reputation scheme that aims to encourage collaboration in OurGrid. Through the autonomous behavior of the peers in the system, the peers who collaborate more are prioritized when they request resources from the community. Being completely autonomous, the Network of Favors has easy implementation and deploying, allowing its use in production in OurGrid without adding significant complexity to its adoption.

Agradecimentos

A Walfredo e Fubica pela orientação dentro e fora do contexto do trabalho da dissertação, companheirismo e bom humor. Sem a disponibildade e abertura de vocês dois para escutar e criticar minhas opiniões, bem como para construir outras, esse trabalho não teria acontecido.

À Professora Patrícia, por ter aceitado me orientar inicialmente no mestrado e ter concordado prontamente com minha decisão de mudar de área.

A Elizeu, companheiro-mór nesses dois anos. Obrigado pela consultoria metodológica, cultural, matemática, festiva e tecnológica.

A Cássio, companheiro durante a maior parte do mestrado na empreitada de morar em Campina Grande.

A Filipe (Pipinho Fenômeno), por ter feito a maior parte do OurGrid sair do papel.

Ao pessoal do Laboratório de Sistemas Distribuídos, especialmente Lauro, Gustavo, Raquel, Lívia, Roberta, Eliane e Glau.

To Miranda, for the ideas, mathematics and cheerfulness .

Também a Hyggo, pelas discussões, músicas e bom humor.

A Keka e Aninha, por terem sido tão prestativas.

À minha Mãe, Marilza, por ter estado sempre perto, apesar dos 122Km de distância.

À ChicoCorrea & ElectronicBand, por ter me mantido na música durante esses dois anos.

Por fi m, parafraseando o Prof. Dalton Guerrero: *à parcela da população brasileira que, sem saber nem poder, financiou minha formação*.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Este capítulo motiva a dissertação. Inicialmente, é discutido o problema tratado e nossos requisitos para uma solução. Em seguida, uma visão geral das soluções existentes mostra porque elas não satisfazem nossos requisitos. Nossa solução é então apresentada em linhas gerais. O capítulo acaba com uma descrição da organização do restante dessa dissertação.

1.1 Motivação

A computação em grid é utilização de recursos distribuídos através de diversos domínios administrativos para compor um serviço [Fos01; FKN+02] que provê uma qualidade de serviço não trivial. Um grid é uma federação de recursos que pode ser acessada por um conjunto de usuários. Essa federação pode ser usada para prover serviços como execução de aplicações paralelas ou armazenamento de grandes quantidades de dados. Também são possíveis serviços de mais alto nível, compostos a partir de serviços como os exemplificados. Contudo, nesta dissertação, consideramos a utilização de grids computacionais apenas objetivando a execução de aplicações paralelas. A possibilidade de agregar um grande número de máquinas em uma plataforma de execução mais barata tem motivado esta utilização da computação em grid, e a maioria dos sistemas que usam grids em produção hoje têm este fim.

Além de focarmos na computação em grid especificamente visando alta performance, nesta dissertação também estamos interessados apenas em sua utilização para executar aplicações *Bag-of-Tasks* (BoT). Aplicações BoT são as aplicações paralelas compostas de tarefas que não se comunicam durante sua execução. As tarefas são independentes e, portanto, essas

aplicações são especialmente adequadas para grid computacionais, onde a largura de banda entre os componentes do sistema pode facilmente ser limitada, já que estes estarão conectados através de uma rede de longa distância. Apesar desta adequação, Cirne et al. [CBS+03] constataram que hoje não existem soluções simples em produção para que os usuários de aplicações BoT usem a computação em grid.

Um dos problemas insuficientemente resolvidos para que os usuários de aplicações BoT possam efetivamente usar a computação em grid é a *composição de grids*. Obter acesso aos recursos para formar um grid não é uma tarefa simples. Não existem soluções largamente adotadas que automatizem a negociação necessária a esse processo. Para compor um grid hoje, é preciso negociar pessoalmente com os donos de todos os recursos que formarão o grid. Após acordar os termos de uso, permissões e prioridades no acesso aos recursos, os provedores de recursos precisam configurar esses recursos para serem acessados pelos usuários. Para que novos usuários acessem esses mesmos recursos, é preciso uma nova negociação e configuração. A necessidade de uma negociação humana e o custo operacional da reconfiguração dos recursos claramente dificultam a composição de grids hoje.

Andrade et al. e Chun et al. argumentam que a razão pela qual não existem soluções em produção hoje para a composição automática de grids é a complexidade da infra-estrutura necessária para viabilizar as propostas existentes [ACBR03; CFV03]. Uma alternativa a essas propostas é a utilização de sistemas par-a-par[1] de compartilhamento de recursos computacionais [ACBR03; CFV03; BZH03]. Embora não resolvam o problema da composição de quaisquer grids, esses sistemas em geral possuem implantação simples, não dependendo de infra-estrutura complexa, e podem ser postos em produção hoje. Estamos desenvolvendo um sistema segundo essa abordagem, a comunidade OurGrid. A comunidade OurGrid — ou simplesmente o OurGrid — é um sistema de compartilhamento par-a-par destinado à composição de grids para a execução de aplicações BoT.

É importante que haja um mecanismo de incentivo à colaboração no OurGrid. Diversos estudos mostraram que em sistemas par-a-par de compartilhamento de arquivos, uma parcela significativa dos usuários não contribui com recursos na comunidade[AH00; RF02; SGG02]. Esses usuários apenas consomem os recursos da comunidade, e são chamados de *free-riders*.

[1] Usamos neste trabalho *par-a-par* como tradução do termo amplamente presente na literatura *peer-to-peer*. Por conseguinte, usamos *par* como tradução de *peer* neste contexto.

Os free-riders podem denegrir o desempenho do sistema para os pares que contribuem com recursos, obtendo recursos em detrimento destes. Considerando que há um custo para a doação de recursos para a comunidade, pode se tornar mais interessante para um par ser um free-rider do que colaborar para o sistema. Quanto menor a quantidade de pares colaborando, menor a quantidade de recursos disponíveis, e maior a contenção de recursos. O custo para doação de recursos representa a largura de banda alocada para prover um arquivo em sistemas de compartilhamento de arquivos, ou o esforço necessário para a instalação e confi guração dos mecanismos de compartilhamento, no caso do OurGrid. É importante que o proveito obtido pelos pares que colaboram com o sistema supere o custo para doar recursos de forma que haja um incentivo para a colaboração na comunidade.

Este trabalho apresenta a Rede de Favores, um mecanismo de incentivo à colaboração para o OurGrid. Esse mecanismo foi desenvolvido com os objetivos do OurGrid de ter fácil implantação e poder ser usado em produção em curto prazo. A Rede de Favores usa um comportamento simples e autônomo nos pares do sistema para a priorização dos colaboradores na comunidade. Essa priorização é feita de forma que o benefício obtido por um par do sistema seja proporcional a quanto ele colaborou no passado. Assim, esperamos estar viabilizando a implantação do OurGrid em curto prazo com um alto nível de colaboração na comunidade.

1.2 Estado da Arte

Motivar a provisão de recursos e regular o acesso a eles são problemas econômicos. Diversas propostas para o problema geral da composição de quaisquer grids são baseadas na criação de uma Economia Grid [ABG02; BV00; BAG00; Gri03], fundada nas soluções bem conhecidas da economia de mercado. Em uma economia grid, usuários que necessitam de recursos negociam seu uso com provedores, e pagam pelo serviço obtido.

Entretanto, essas soluções visam à criação de grids onde qualquer tipo de aplicação pode ser executada. Esse é um problema bem mais complexo do que a composição de grids apenas para aplicações BoT. Para a execução de uma aplicação que necessite de comunicação intensa entre as tarefas, por exemplo, o cliente precisará da alocação de um conjunto de recursos que atenda a um nível de qualidade de serviço (QoS). O usuário precisará tanto que

as máquinas que tenham uma determinada capacidade de processamento e disponibilidade quanto que seu canal de comunicação possua um mínimo de largura de banda disponível. Exigências deste tipo complicam a implantação das soluções que viabilizam a criação de grids genéricos, ou seja, grids para qualquer tipo de aplicação. Essas exigências também tornam necessárias a auditoria do serviço prestado pelo provedor, pois um cliente precisa de garantias que só pagará o valor do serviço realmente obtido. Os provedores, por outro lado, precisam de garantias de que os clientes pagarão este valor, tornando necessários também mecanismos de moedas e bancos eletrônicos.

As tecnologias necessárias para viabilizar o uso de mecanismos de dinheiro e banco eletrônicos e a auditoria de serviços computacionais não estão disponíveis comumente e não é uma tarefa trivial torná-las altamente disponíveis. Assim, acreditamos que não existem soluções baseadas na economia grid em produção hoje devido à complexidade da infra-estrutura necessária e à sua indisponibilidade.

A ausência de mecanismos mais simples do que os baseados na economia grid impede usuários de aplicações mais simples, como as BoT, de usufruir da computação em grid hoje. Acreditamos que os usuários de aplicações deste tipo podem tirar proveito da computação em grid hoje através de soluções que se baseiem na simplicidade das aplicações BoT e tenham fácil implantação em cenários reais.

1.3 Objetivos

O objetivo desse trabalho é apresentar uma solução que incentive a colaboração no cenário da comunidade OurGrid. Esperamos com isso facilitar a composição de grids computacionais para usuários de aplicações BoT em curto prazo. Nossa solução deve atender aos seguintes requisitos:

1. Incentivar à colaboração na composição de grids computacionais de escala arbitrária para a execução de aplicações BoT;

2. Viabilizar uma implementação em curto prazo; e

3. Possuir uma implantação simples.

O requisito 1 define que nossa solução deve ser uma alternativa às propostas existentes no escopo das aplicações BoT. Os requisitos 2 e 3 especificam que ela não deve depender de infra-estruturas para sua implantação que inviabilizem seu uso a curto prazo. Nossa intenção é, delimitando um sub-problema da questão geral de composição de grids computacionais, conceber uma solução eficiente e simples o suficiente para viabilizar sua utilização em produção hoje. Acreditamos não ser necessária a criação de uma economia grid para beneficiar os usuários de aplicações BoT.

1.4 A Rede de Favores

Nossa solução para incentivar a colaboração em um sistema par-a-par é utilizar um esquema de reputação. A reputação de um participante do sistema reflete o quanto ele já colaborou para a comunidade, e esta prioriza os pares de acordo com suas reputações. Assim, todos têm um incentivo para colaborar.

Neste trabalho, propomos um novo esquema de reputação, chamado de *Rede de Favores*. Diversos esquemas de reputação para sistemas par-a-par já foram propostos na literatura [DdVPS03; KSGM03; OLK03], e a Rede de Favores se diferencia dos esquemas propostos por utilizar apenas o comportamento autônomo dos pares da comunidade para priorizar os colaboradores. Nosso resultado principal é a constatação de que é possível, no cenário do compartilhamento de recursos computacionais entre usuários de aplicações BoT, incentivar eficientemente a colaboração usando um esquema de reputação autônomo.

Em um esquema de reputação autônomo, não há transitividade nas opiniões dos pares sobre a reputação de um par. A comunidade não tenta obter um valor que represente a opinião de seus integrantes para priorizar um determinado par. Em lugar disto, cada par no sistema prioriza os demais de acordo com suas próprias interações passadas com os outros pares. Assim, não há a necessidade de um protocolo para a obtenção desta informação nem a preocupação com a validade da informação obtida. A necessidade de garantir a validade é importante uma vez que pode ser do interesse de alguns pares mentir sobre essa informação. Em outros sistemas, mecanismos como infra-estruturas criptográficas, votação e tabelas hash distribuídas são necessárias para assegurar essa validade [DdVPS03; KSGM03].

No Capítulo 2, mostramos que a Rede de Favores provê o incentivo à colaboração dos

pares no cenário da comunidade OurGrid. Este comportamento, somado à simplicidade da sua implementação e implantação satisfazem os nossos requisitos. Para implementar a Rede de Favores em uma comunidade par-a-par basta introduzir em cada par um comportamento autônomo extremamente simples, e sua implantação depende apenas de um mecanismo de contabilização do trabalho obtido de um outro par, algo necessário a qualquer esquema de reputação.

1.5 Organização da Dissertação

A banca examinadora deste é composta por examinadores de língua portuguesa e inglesa. Para viabilizar o exame por todos os orientadores, a dissertação é composta de um texto em Português que apresenta o cerne das contribuições de um conjunto de apêndices em Inglês que explora aspectos dessas contribuições.

No Capítulo 2, discutimos a Rede de Favores e apresentamos o resultado principal desta dissertação: o de que é possível incentivar a colaboração em uma comunidade de compartilhamento de recursos computacionais para rodar aplicações BoT usando um esquema de reputação autônomo.

O Capítulo 3 contextualiza nosso trabalho, apresentando o estado da arte da área. Apresentamos a forma de compor grids usada em produção hoje, as propostas existentes para economias grid e outros esquemas de reputação par-a-par relacionados à Rede de Favores.

O Capítulo 4 é um guia para a leitura dos apêndices. Cada um dos três apêndices investiga em mais detalhes um aspecto das contribuições da dissertação, na forma de um artigo científico. Esses artigos científicos foram publicados ou submetidos a conferências internacionais e são, por conseguinte, escritos em Inglês. Apesar de terem sido escritos em colaboração com outros autores, todos os artigos incluídos como apêndices têm como primeiro autor o autor desta dissertação.

Acreditamos que esses apêndices refletem as contribuições feitas ao estado da arte de grids computacionais oriundas do trabalho que culminou com esta dissertação. Os apêndices apresentam não apenas nossos resultados, mas também a evolução do trabalho. Por fim, sendo escritos em Inglês, tornam o trabalho acessível em detalhes para o examinador de língua inglesa que faz parte da banca examinadora da dissertação.

Por fi m, no Capítulo 5 contém um apanhado de nossas conclusões e dos trabalhos futuros que vemos possíveis a partir de nossas contribuições.

Capítulo 2

Reputação Autônoma no Compartilhamento de Recursos Computacionais

Este capítulo explica o funcionamento da Rede de Favores, o esquema de reputação autônoma concebido neste trabalho, e mostra que ela incentiva a colaboração em uma comunidade que compartilha seus recursos computacionais para rodar aplicações BoT. A Seção 2.1 discute o cenário do problema considerado. Na Seção 2.2 mostramos analiticamente que se o sistema considerado tem um mecanismo que identifica os colaboradores com um certo nível de precisão e os colaboradores conhecidos têm prioridade no acesso aos recursos, então existe um incentivo para que um par colabore. Na Seção 2.3 o funcionamento da Rede de Favores é discutido em detalhes. A Seção 2.4 mostra através de simulações que a Rede de Favores satisfaz as condições expostas na Seção 2.2 e, assim, incentiva à colaboração no cenário proposto. Por fim, a Seção 2.5 discute como a Rede de Favores atende aos requisitos da simplicidade de implementação e implantação. Esta seção discute também as limitações práticas da Rede de Favores.

2.1 Definição do Problema

Nesta seção, discutimos o cenário do problema considerado. Especificamente, delimitamos nosso problema e expomos todas as considerações que fizemos na obtenção de nossos resul-

tados.

Nós consideramos um sistema par-a-par formado por um conjunto de sites que compartilham seu poder computacional ocioso. Cada par do sistema é um site, e possui como recursos um conjunto de máquinas. Cada site possui também um conjunto de usuários que as utilizam periodicamente para executar suas aplicações Bag-of-Tasks. Esses usuários geram, no par, em alguns momentos, demanda por mais recursos do que este possui. Quando isso acontece, o par requisita recursos à comunidade. Quando não possui demanda por recursos, cada par torna disponível um subconjunto de seus recursos ociosos para a comunidade.

Consideramos que existe um custo para doar recursos para a comunidade. Esse custo representa, por exemplo, a necessidade da instalação e configuração do software para compartilhar uma máquina na comunidade, riscos de segurança no site ou o tempo necessário para liberar a máquina caso ela esteja sendo usada pela comunidade e um usuário local a requisite. O custo para doação modela a motivação para a não-contribuição na comunidade.

Neste cenário, dizemos que um par *colabora* com o sistema quando ele doa recursos para a comunidade. Quanto maior a quantidade de recursos doados, mais um par colabora com o sistema. Chamamos de *free-rider* o par que não doa recurso algum, e de *free-riding* esta atitude. Não consideramos nesse trabalho outros comportamentos, como o de pares mal-intencionados.

Desejamos mostrar que, utilizando a Rede de Favores, há um incentivo para que os pares colaborem. Há um valor de utilidade positivo ganho por um par ao consumir recursos e um negativo ao doá-los. Ao longo do tempo, o valor da utilidade de um par sendo colaborador deve ser maior que o valor de sua utilidade sendo free-rider. O lucro subjacente à colaboração deve compensar o custo da doação de recursos.

Para nosso estudo desse cenário, fizemos três considerações:

1. Alguns membros da comunidade têm, em alguns períodos, recursos ociosos.

2. As aplicações que serão usadas pelos clientes da comunidade não necessitam de garantias de qualidade de serviço (QoS).

3. Os clientes da comunidade são *consumidores ávidos*. Isso significa que quando eles possuem demanda por recursos, eles podem se beneficiar do uso de quaisquer recursos

disponíveis. Os clientes da comunidade obtém uma utilidade positiva do uso tanto de qualquer tipo quanto de qualquer quantidade de recursos.

Acreditamos que essas considerações condizem com as características do problema que estamos considerando e, com base nelas, estabelecemos algumas simplificações para o problema geral de composição de grids que viabilizam o uso da Rede de Favores de forma efetiva.

A consideração de que alguns membros da comunidade possuem recursos que ficam ociosos parte de seu tempo reflete o ciclo normal do uso de recursos computacionais para computação de alto desempenho. Nossa experiência mostra que o usuário normalmente (a) planeja sua computação, (b) executa uma aplicação, (c) examina os resultados obtidos e (d) reinicia o ciclo. Freqüentemente, uma quantidade significativa de tempo é necessária para o planejamento dos parâmetros da computação e para o entendimento de seus resultados. Durante esse tempo, o usuário não utiliza seus recursos. Mesmo que os recursos de um site sejam utilizados por vários usuários, consideramos que haverão períodos de ociosidade nesses recursos. A existência desses recursos ociosos viabiliza a formação de uma comunidade de compartilhamento baseada em trocas. Cada integrante do sistema doa seus recursos ociosos para comunidade objetivando ser recompensado quando necessitar dos recursos compartilhados.

Note que se o principal recurso compartilhado é poder de processamento, os recursos não têm valor quando ociosos, e não podem ser armazenados para utilização posterior. Na Figura 2.1, ilustramos dois sites de usuários de aplicações BoT compartilhando seus recursos computacionais. Durante os períodos em que não há usuários usando os recursos do site, estes são doados para a comunidade. Quando há demanda por recursos no site, este é capaz de obter da comunidade recursos extras. Trocando seus recursos ociosos pelo acesso a recursos da comunidade quando necessário, o site tem disponível uma parte do poder computacional que ele não utilizou e não tem como armazenar.

Nossa segunda consideração baseia-se no fato das aplicações BoT não necessitarem de garantias de QoS para sua execução. Se uma máquina que estava executando uma tarefa torna-se indisponível, a tarefa pode ser simplesmente submetida outra vez, sem impacto para as demais já em execução. A independência de garantias de QoS torna possível dispensar as negociações por recursos na comunidade, bem como mecanismos de auditoria. Não é

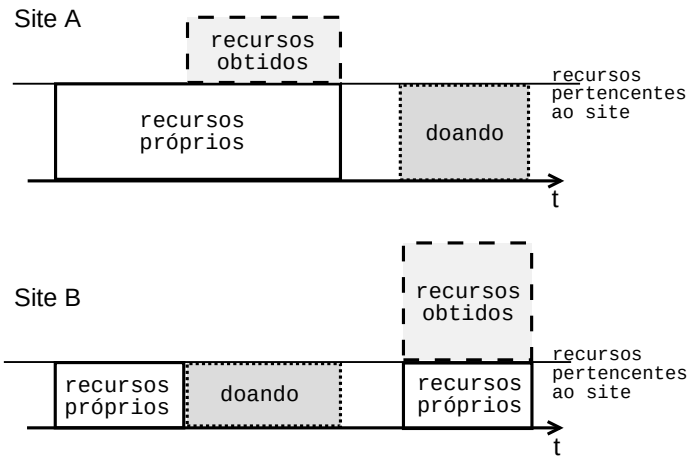


Figura 2.1: Compartilhamento de recursos entre dois sites de usuários de aplicações BoT.

necessário especificar níveis de QoS nem verificar se eles foram atendidos para utilizar um recurso, uma vez que as aplicações podem se beneficiar de quaisquer recursos obtidos.

A terceira consideração implica que modelamos a necessidade por recursos de um usuário de aplicação BoT como sendo infinita durante um período de tempo. Inicialmente, caso um usuário não necessitasse de uma quantidade significativa de recursos, não seria necessário para ele utilizar a computação em grid. Além disso, a replicação das tarefas em outros processadores é uma forma simples de diminuir o tempo necessário para a execução de uma aplicação BoT [PCB03]. Esta replicação multiplica a demanda de um usuário, e potencialmente o torna capaz de utilizar qualquer quantidade de recursos que ele tenha disponível.

Além da capacidade de consumir qualquer quantidade de recursos, a terceira consideração define que quando um integrante do sistema possui demanda por recursos, ele pode ser atendido pelos recursos de qualquer outro integrante. Consideramos, portanto, que temos apenas um bem sendo compartilhado na comunidade. O nosso conceito de poder computacional engloba todos os recursos sendo compartilhados, como espaço em disco, memória, processador e largura de banda, e todos têm a mesma disponibilidade. Para nosso estudo não consideramos o efeito que a escassez de algum desses bens poderia causar na comunidade. Essa avaliação é um dos trabalhos futuros identificados a partir de nossos resultados. Note que não sabemos se no cenário do OurGrid existirão recursos mais escassos e requisitados. Nossa experiência mostra que o principal recurso necessário aos usuários de aplicações BoT é CPU. Além disso, que eles podem, em geral, recompilar suas aplicações para as platafor-

mas que têm disponíveis. Essa experiência sugere que eles podem usar a grande maioria dos tipos de recursos disponíveis.

No cenário descrito e com as considerações expostas, nosso problema é incentivar a colaboração dos pares da comunidade. Desejamos mostrar que a Rede de Favores provê esse incentivo através do comportamento autônomo dos colaboradores. Identifi camos como pior caso deste problema considerar nosso sistema par-a-par inicialmente sem informação sobre seus participantes e composto por uma parcela de free-riders. A partir deste caso mostraremos que usando a Rede de Favores no cenário descrito (i) os free-riders são marginalizados, possuindo uma utilidade obtida do sistema menor que a dos colaboradores, e (ii) há eqüidade na divisão dos recursos disponíveis entre os colaboradores. Por eqüidade, denotamos que os pares que doam mais recebem mais recursos da comunidade.

2.2 Compartilhamento de Recursos Computacionais Entre Consumidores Ávidos

Nessa seção, provamos um resultado analítico sobre o compartilhamento de recursos entre consumidores ávidos: se a comunidade tem algum mecanismo (não necessariamente a Rede de Favores) através do qual ela identifi ca os colaboradores com precisão sufi ciente e os colaboradores conhecidos têm prioridade no acesso aos recursos, então existe um incentivo para que um par colabore. Como conseqüência, se os pares mudam sua estratégia de free-riding para colaboração e vice-versa de acordo com seu interesse, então a comunidade evolui para um estado onde não há free-riders.

Seja n o número de pares na comunidade. Como é possível que os pares mudem suas estratégias, o número de free-riders variará no tempo. Seja $f(t).n$ o número de pares que são free-riders no tempo t. Os outros $(1 - f(t)).n$ pares são colaboradores no tempo t, doando todos os seus recursos ociosos para a comunidade.

Assumimos que a utilidade perdida por um doador pela doação de um recurso é um múltiplo fi xo v da utilidade ganha pelo consumidor do recurso, com $0 < v < 1$. O custo deve ser maior que zero para que haja motivação para a não-contribuição e menor que 1 para que haja incentivo à doação de recursos necessária para o compartilhamento de recursos. Acreditamos que, na prática, esse custo deve ser um valor pequeno, pois vemos o benefício obtido

com o acesso a quantidades signifi cativas de recursos como sendo bem maior que o esforço para a doação de recursos. Acreditamos que esse esforço será pequeno pois compreenderá apenas a instalação de uma infra-estrutura leve para o compartilhamento de recursos, considerando que esta é segura e garante que os usuários locais não terão prejuízos no acesso aos seus recursos compartilhando-os com a comunidade, fatores que pretendemos garantir no OurGrid.

Se $f(t) = 0$, então todos os pares são colaboradores. Se $f(t) = 1$, então a comunidade contém apenas free-riders, e não há recursos para serem doados ou incentivo para par algum permanecer na comunidade. Suponha agora que $0 < f(t) < 1$ em algum tempo t. Defi nimos $\epsilon(t)$ como a probabilidade de que um recurso seja doado para um free-rider no tempo t. Esse "erro" pode acontecer tanto porque um ou mais colaboradores não conseguem distinguir outro colaborador para doar recursos ou porque há apenas free-riders requisitando recursos no tempo t. Se não há colaboradores no tempo t, defi nimos $\epsilon(t) = 1$.

Se a utilidade ganha pelo consumidor de uma doação de um recurso é u, então o ganho de utilidade total esperado pelo conjunto de colaboradores como o resultado de uma doação é $(1 - \epsilon(t)).u - v.u$, onde t é o tempo em que a doação acontece. Note que o doador deve ser um colaborador, uma vez que free-riders não doam recursos. O ganho total de utilidade esperado para o conjunto dos free-riders é $\epsilon(t).u$. Como há $(1 - f(t)).n$ colaboradores, o ganho médio esperado de utilidade para um colaborador é $\frac{(1-\epsilon(t)-v).u}{(1-f(t)).n}$. De forma análoga, como há $f(t).n$ free-riders, o ganho médio esperado para um free-rider é $\frac{\epsilon(t).u}{(f(t).n)}$. Assim, é mais proveitoso ser um colaborador se $\frac{(1-\epsilon(t)-v).u}{(1-f(t)).n} > \frac{\epsilon(t).u}{(f(t).n)}$, o que acontece se e somente se $\epsilon(t) < (1 - v).f(t)$. Por outro lado, é mais proveitoso ser um free-rider quando $\epsilon(t) > (1 - v).f(t)$. Quando $\epsilon(t) = (1 - v).f(t)$, free-riders e colaboradores têm a mesma utilidade esperada do sistema. Portanto, se $0 < f(t) < 1$ e $\epsilon(t)$ é menor que $(1 - v).f(t)$, então existe um incentivo para a colaboração.

Nós assumimos que os pares mudarão gradativamente suas estratégias para free-riding ou colaboração se for de seu interesse fazê-lo, ou seja, se a utilidade esperada com a nova estratégia for maior que a esperada com a estratégia atual. Por conseguinte, se há um t' para o qual $\epsilon < (1 - v).f(t)$ para todo $t \geq t'$, então após o tempo t' será sempre do interesse dos free-riders tornarem-se colaboradores. Segue que o free-riding se extinguirá da comunidade.

2.3 A Rede de Favores

Na Rede de Favores, alocar um processador para um membro da comunidade que o requisitou é um favor, e o valor desse favor é o valor do trabalho feito nesse processador pelo requisitante. Cada par mantém um registro local do valor total dos favores que ele fez e recebeu no passado para cada outro par conhecido. Cada vez que um par faz ou recebe um favor, ele atualiza esse registro. Cada par calcula uma reputação local para cada outro par baseado nesse registro, de forma que um par que tenha feito muitos favores e recebido de volta poucos tenha uma alta reputação. Ao contrário, pares que receberam muitos favores e retribuíram poucos têm uma reputação baixa. O par então usa essa reputação local para decidir para quem prestar um favor quando ele tem que arbitrar entre diversos requisitantes.

Como ilustração, suponha que os pares A, B e C nunca interagiram antes. Seja $r_A(B)$ a reputação de B segundo A. Considerando $v(B, A)$ o valor dos favores prestados por B a A, uma forma simples de definir $r_A(B)$ é $r_A(B) = v(B, A) - v(A, B)$. Assim, temos inicialmente $r_A(B) = r_A(C) = 0$. Se B doar alguns recursos para A por algum tempo, o valor de $r_A(B)$ crescerá (enquanto $r_A(C)$ será ainda zero). Se, então, A tem que escolher entre uma requisição de B e uma de C, A escolherá prestar um favor para B. A tomaria a mesma decisão se C também tivesse interagido com ele anteriormente, mas $r_A(C)$ fosse menor que $r_A(B)$.

Note que a reputação de um dado par será diferente aos olhos de diferentes pares, e não há esforço da comunidade para deduzir um valor global para essa reputação. Ainda assim, como veremos na Seção 2.4, o comportamento emergente do sistema quando todos os seus integrantes usam o comportamento autônomo descrito é priorizar os pares com maior reputação sempre que há contenção de recursos. Essa prioridade é o incentivo à colaboração na comunidade.

Esse uso de reputação difere de outros sistemas, onde a reputação sobre os provedores é utilizada pelos consumidores de recursos para aumentar a qualidade de suas transações [eBa03] ou para marginalizar pares suspeitos [OLK03]. Na Rede de Favores o sistema recompensa os pares que contribuíram mais. A idéia é que essa recompensa supere o custo para a doação de recursos para a comunidade.

Na rede de favores, os pares que colaboram com o sistema nunca se recusam a doar

recursos. Qualquer recurso disponível e não disputado pode ser acessado por qualquer par do sistema. O incentivo à colaboração vem apenas da prioridade dada aos colaboradores, e essa prioridade acontece apenas quando há contenção de recursos. A ausência de contenção signifi ca que há recursos para satisfazer a todos, e o fato de os pares conseguirem acessar recursos sem necessariamente doar antes resolve o problema de iniciar o crédito na comunidade. Caso isso não acontecesse, os pares precisariam de algum outro mecanismo de ganho de crédito inicial ao entrar na comunidade, o que adiciona complexidade à estrutura necessária para a Rede de Favores e vai de encontro aos nossos requisitos.

2.3.1 Cálculo da Reputação de um Par

Na Rede de Favores, o par A calcula $r_A(B)$, o valor da reputação local do par B, usando duas informações: o valor dos favores que A recebeu de B e o valor dos favores que B recebeu de A.

Seja $v(A, B)$ o valor total dos favores prestados pelo par A ao par B no passado. Queremos que $r_A(B)$ seja uma função de $v(A, B)$ e $v(B, A)$, e que o valor dessa função cresça quando B faz um favor a A, decresça quando A faz um favor a B, e que seja zero se A nunca interagiu com B.

A função mais simples de $v(A, B)$ e $v(B, A)$ que satisfaz essas condições é:

$$r_A(B) = v(B, A) - v(A, B) \tag{2.1}$$

Essa função é o saldo dos favores que B fez para A. No restante deste trabalho, mostraremos que a Rede de Favores é capaz de incentivar a colaboração em uma comunidade de compartilhamento de recursos computacionais usando esta função. Aprimoramentos podem ser feitos nela para, por exemplo, torná-la robusta a alguns comportamentos maliciosos. Contudo, esses aprimoramentos não são necessários para a obtenção de nosso resultado com as considerações discutidas. Dois aprimoramentos possíveis são discutidos no Apêndice B.

2.4 Avaliação

Para mostrar que a Rede de Favores incentiva a colaboração no cenário considerado no nosso problema, usamos simulações. Nestas simulações, objetivamos capturar o cerne do funcio-

namento da Rede de Favores. Assim, desprezamos diversos aspectos da simulação de uma rede par-a-par, como topologia e falhas, por exemplo. Consideramos uma malha de pares onde todos podem se comunicar entre si e esta comunicação se dá sem falhas. O sistema é composto de 100 pares, a linha do tempo é dividida em turnos, e em cada turno um par pode possuir demanda pelos recursos da comunidade com uma probabilidade ρ.

Inicialmente, na Seção 2.4.1, consideramos colaboradores que contribuem com todos os seus recursos e free-riders apenas. Com essa consideração, mostramos através de simulações e com base na avaliação analítica exposta na Seção 2.2 que há incentivo à colaboração. Em seguida, na Seção 2.4.2, consideramos diferentes níveis de contribuição para mostrar, através de simulações, que os pares que doam mais recebem mais recursos em troca. O segundo resultado mostra que, entre os colaboradores, há incentivo à maior contribuição possível.

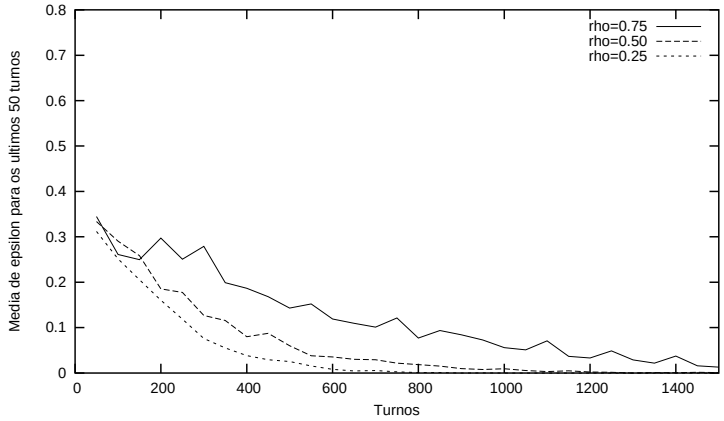
2.4.1 Incentivo à Colaboração

Inicialmente, consideramos que a colaboração é binária: um par colabora com todos os seus recursos ou com nenhum. Dos 100 pares, $(1 - f).100$ são *colaboradores* e $f.100$ são *free-riders*. Quando não possuem demanda por recursos da comunidade, os colaboradores doam todos os seus recursos. Os free-riders, por outro lado, nunca doam. Quando não consumindo, eles fi cam ociosos. Apesar de aqui apresentarmos apenas os resultados para comunidades de 100 pares, nossa investigação mostrou que estes não diferem signifi cativamente dos resultados para comunidades de 500 ou 1000 pares.

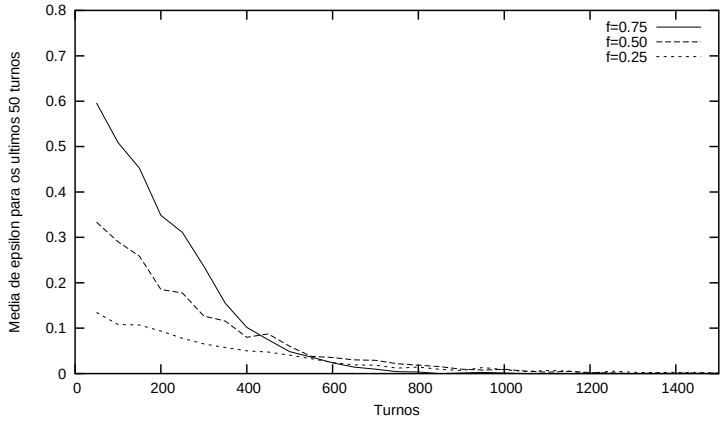
Para estimar $\epsilon(t)$, medimos a fração dos recursos da comunidade que foi doada para free-riders no tempo t. Desejamos mostrar que uma comunidade de compartilhamento de poder computacional com as considerações discutidas na seção anterior é capaz de identifi car e marginalizar os free-riders utilizando a Rede de Favores.

A Figura 2.2 mostra a medição para a simulação de uma comunidade de 100 pares com diferentes valores para f e ρ. Como $\epsilon(t)$ varia muito entre turnos, usamos a média de $\epsilon(t)$ dos últimos 50 turnos para suavizar a curva no gráfi co. É possível ver na fi gura que com o passar do tempo, a comunidade identifi ca os free-riders (ou seja, $\epsilon(t)$ fi ca muito pequeno) e eles não conseguem mais recursos.

As Figuras 2.2(a) e 2.2(b) detalham o comportamento da Rede de Favores com a variação de ρ e f, respectivamente. O tempo necessário para que a comunidade atinja o estágio onde



(a) $f = 0.5$ e diferentes valores de ρ



(b) $\rho = 0.5$ e diferentes valores de f

Figura 2.2: Medição de $\epsilon(t)$ para diferentes valores de f e ρ

os free-riders já foram identificados suficientemente para que $\epsilon(t)$ seja desprezível depende de ρ. Isso acontece porque a comunidade distingue os colaboradores quando eles doam, e quando mais alto o valor de ρ, menos freqüentemente os colaboradores doam. Em outras palavras, quanto mais ρ se aproxima de 1, mais o comportamento dos colaboradores e free-riders é similar, e mais tempo é necessário para que a comunidade os distinga.

Por sua vez, a variação nos valores de f não afeta de forma significante o tempo necessário para que a comunidade identifique os free-riders. Essa variação afeta os valores de $\epsilon(t)$ antes que essa identificação aconteça. Quando mais alto f, maiores os valores das medições de $\epsilon(t)$ antes que a comunidade identifique os free-riders, pois quando um colaborador doa

seus recursos para um par sobre o qual ele não tem informação ainda, a probabilidade de que esse par seja um free-rider é proporcional a f.

Nestas simulações consideramos que os pares não mudam suas estratégias de free-riding para colaborar ou o contrário. Se eles mudassem de estratégia, o valor de f não seria fixo, e variaria de acordo com o número de free-riders. Entretanto nossas simulações mostram que para diferentes valores de f o sistema tende a um estado onde free-riders não obtém recursos. Se os colaboradores estão em vantagem ou não depende do custo de doação v. Os colaboradores estão em vantagem se $\epsilon < (1-v).f$ (de acordo com o resultado da Seção 2.2), temos é mais proveitoso ser um colaborador se $v < 1 - \epsilon/f$. Em todos os cenários investigados, $\epsilon(t)$ tende a um valor muito próximo de zero com o tempo. Em particular, $\epsilon(t)$ fica abaixo de $1/2$ após um intervalo suficiente de tempo. Então, mesmo para um valor alto de f, como $f = 0.75$, por exemplo, a melhor estratégia é colaborar se $v < 1/3$. Esperamos que o OurGrid tenha um custo de instalação pequeno, dada sua independência de infra-estruturas complexas e à possibilidade de cancelar uma tarefa de um usuário da comunidade para dar prioridade a usuários locais. Assim, esperamos que, no OurGrid, o custo de doação v seja próximo de zero (e certamente menor que $1/3$). Note que se ϵ ou f forem menores, a melhor estratégia é colaborar mesmo que o custo de doação seja maior que $1/3$.

Assim, os resultados de nossas simulações mostram que se os pares de um sistema de compartilhamento de recursos para aplicações BoT como o considerado usarem a Rede de Favores e mudarem suas estratégias de acordo com seu interesse econômico, o free-riding na comunidade se extinguirá.

2.4.2 Eqüidade

Nossos resultados até agora mostram que quando há contenção de recursos, a Rede de Favores dificulta que pares que não colaboram com a comunidade acessem os recursos compartilhados por ela. Mostramos agora que, entre os colaboradores, aqueles que doam mais recebem mais recursos em troca. Assim, é do interesse dos pares na comunidade contribuírem o máximo possível de seus recursos. Para isso, introduzimos pares que têm diferentes quantidades de recursos. Nos resultados apresentados nesta seção, a quantidade de recursos possuída por cada par é dada pela distribuição uniforme $U(1, 19)$. Quando o par não está consumindo, todos os seus recursos são doados.

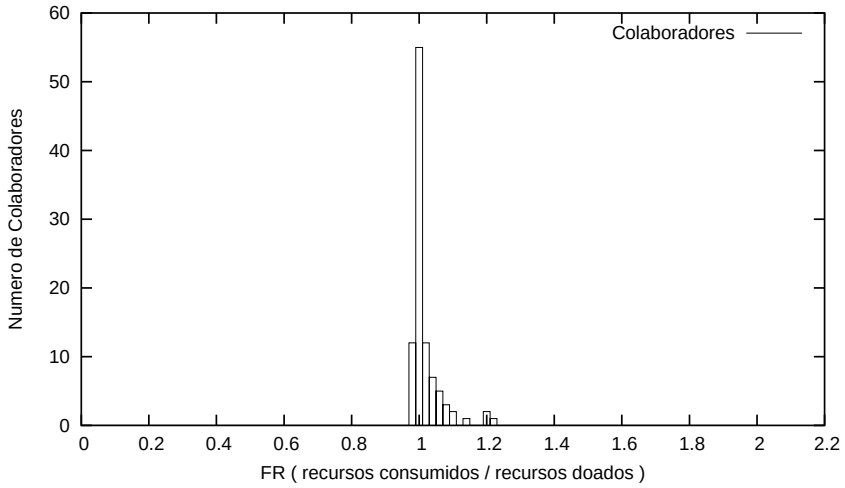


Figura 2.3: Histograma de FR para uma comunidade de 100 colaboradores com $\rho = 0.5$ e cuja quantidade de recursos possuída é dada pela distribuição uniforme $U(1, 19)$ após 3000 turnos

Para mostrar que os pares se beneficiam doando mais recursos, nós definimos uma segunda métrica: a razão de favores, que chamamos FR. Definimos FR como a razão entre o valor dos recursos obtidos e doados. Assim, se temos contenção de recursos e $FR = 1$, temos que a comunidade promove eqüidade no compartilhamento de seus recursos. A eqüidade significa que os pares têm como retorno da comunidade uma quantidade de recursos proporcional a sua doação. Havendo contenção sempre de recursos, os pares devem ser capazes de obter de volta o mesmo que doaram, ou seja, devem ter $FR = 1$.

A Figura 2.3 mostra a distribuição dos valores de FR em uma comunidade de 100 colaboradores com $\rho = 0.5$, após 3000 turnos. Optamos por usar um histograma para ilustrar FR para dar uma visão dos valores assumidos devido ao número de pares da comunidade, que torna inviável a apresentação dos valores de todos os pares no tempo. É possível ver nesta figura que após 3000 turnos FR está próximo de 1 para todos os pares, o que, como discutido, denota eqüidade.

Alterações nos valores de ρ e f não afetam de forma significante o histograma dos valores de FR da comunidade. Como há sempre contenção de recursos, e os recursos são distribuídos entre os colaboradores apenas. Nestas condições, a Rede de Favores é capaz de sempre manter a eqüidade. A quantidade de turnos considerada na simulação, entretanto,

afeta o histograma: com mais turnos, os valores de FR convergem ainda mais para 1.

Esse resultado mostra que a Rede de Favores incentiva a contribuição da maior quantidade de recursos possível por parte de um par. Como todos os consumidores são ávidos, eles podem se beneficiar de todos os recursos obtidos. Além disso, o custo para doação v é uma fração da utilidade obtida com o consumo de recursos. Portanto, com $FR = 1$, quanto maior for a quantidade de recursos doados, maior é a utilidade obtida por um par do uso do sistema.

2.5 Considerações

A Rede de Favores satisfaz os nossos requisitos para um esquema de reputação: ela incentiva a colaboração de recursos para a comunidade e possui implementação e implantação simples.

Para os casos considerados, ela provê o incentivo à colaboração sem depender de um valor global de reputação para um par. A colaboração é promovida usando o comportamento autônomo dos integrantes do sistema, que usam apenas informações locais para tomar suas decisões. Este comportamento é simples, não adicionando complexidade significativa à implementação de uma comunidade de compartilhamento par-a-par.

Além disso, a Rede de Favores não depende de nenhuma infra-estrutura para garantir a integridade da informação obtida da comunidade sobre interações entre um par e terceiros, como infra-estrutura criptográfica, auditoria ou infra-estrutura especializada de armazenamento (como tabelas hash distribuídas). Em outras propostas, é necessária uma infra-estrutura deste tipo para garantir a integridade da informação extraída da comunidade sobre a reputação de um par. Na Rede de Favores a reputação de um par será tipicamente diferente aos olhos de diferentes pares, baseado em suas interações passadas, mas não há tentativa de combinar esses diferentes valores locais em uma visão global.

A Rede de Favores depende apenas de um mecanismo: o de obtenção de informação confiável sobre os valores dos favores prestados pelos pares. Especificamente, nós consideramos que um par pode medir o valor de um favor feito por outro par. Essa medição inclui a verificação de que o resultado do trabalho retornado pelo par provedor é valido, e não dados falsificados. Neste caso, o favor não teria valor. Note que a consideração da capacidade de valorar os serviços obtidos é comum aos trabalhos do estado da arte em esquemas de

reputação par-a-par.

Pretendemos atacar o problema da valoração dos favores como trabalho futuro. É importante que a solução para esse problema seja robusta a valorações maliciosas de outros pares. Para lucrar, um par mal-intencionado pode valorar seus favores como valendo mais que seu valor justo, ou sabotar seus consumidores. Sabotar significa retornar resultados falsos para as tarefas recebidas sem executá-las para receber o crédito pelo favor forjado. Além de tolerar esses comportamentos, temos como requisito para nossa solução que ela seja de fácil implementação e implantação, e que possa ser incorporada à comunidade OurGrid.

Uma possibilidade identificada é um par A, utilizar replicação das tarefas executadas como favores no par B em outros pares para verificar o valor do favor reportado por B. Assim, alguns recursos adicionais são consumidos para a verificação dos valores retornados. Sarmenta [Sar02] propôs uma abordagem similar visando à descoberta de sabotagem (a não execução correta de uma tarefa) em sistemas de computação voluntária. Nesta abordagem, chamada *tolerância à falhas baseada em credibilidade* o cliente envia cada tarefa que ele deseja executar para diferentes provedores de serviço até que pelo menos um número determinado de resultados retornados seja igual. A comparação dos resultados atesta a validade deles. Agindo corretamente, um provedor ganha credibilidade na visão de um consumidor, e o consumidor ganha confiança sobre seus resultados. A freqüência com que a replicação das tarefas em B é feita diminui quando B ganha credibilidade. Com menos replicação, diminui o desperdício de recursos para a verificação de recursos. Por fim, em lugar da execução real pequenas tarefas-sonda cujo resultado é previamente conhecido podem ser enviadas para verificar o comportamento dos provedores de serviço. O uso dessas tarefas diminui ainda mais o desperdício de recursos.

Suspeitamos que em um sistema usando a Rede de Favores, o mecanismo baseado em credibilidade proposto por Sarmenta poderia ser utilizado para verificar a sabotagem e também a valoração dos favores feita pelos pares. Essa abordagem nos parece adequada por não introduzir componentes centralizados ou infra-estruturas complexas no sistema. Além disso, ela permite uma diminuição na quantidade de recursos desperdiçada em relação a mecanismos de voto simples. Note que a implementação de um sistema de tolerância à sabotagem dos resultados retornados por outros pares é necessária em qualquer sistema de compartilhamento de recursos com partes não confiáveis.

Capítulo 3

Trabalhos Relacionados

Neste capítulo situamos nosso trabalho em relação ao estado da arte em composição de grids computacionais e esquemas de reputação par-a-par.

3.1 Compondo Grids circa 2003

Embora hoje existam tecnologias para a utilização de grids computacionais, não existem soluções amplamente adotadas para criá-los. Embora existam propostas para viabilizar a negociação automática de acesso a recursos disponibilizados em um grid, na prática, o mecanismo utilizado ainda é a negociação humana. Tanto para formar grids entre instituições quanto para que usuários congreguem todos os recursos a que tem acesso em um grid, é necessária a negociação do acesso com os responsáveis pelos recursos.

O Globus Toolkit [FK97] é a o padrão de facto para computação em grids hoje. O Globus abrange todas as ferramentas necessárias para compor um grid. Entretanto, essa composição não é automática. Suponha que várias instituições diferentes queiram formar uma federação. É necessário que todas essas instituições negociem as permissões e prioridades no acesso aos recursos compartilhados. Essa negociação é totalmente não-automatizada. Uma situação similar acontece em outras abordagens. O APST [COBw00] e o MyGrid [CPC+03] são duas ferramentas semelhantes para o uso da computação em grids. Ambos se propõem a fornecer para seus usuários acesso a recursos que possuam diversos tipos de infra-estrutura, como Globus, Condor ou acesso via *Secure Shell* (SSH). Assim, um usuário pode compor um grid a partir de todos os recursos a que ele tem acesso, não dependendo de infra-estruturas espe-

cífi cas. Cirne et al. [CBS+03], entretanto, relatam que é difícil para um usuário conseguir acesso a recursos em mais que uma pequena quantidade de sites cujos responsáveis o usuário conhece.

O projeto Condor [LLM88] surgiu originalmente como uma forma de acessar os recursos computacionais ociosos de uma instituição. Epema et al. [ELvD+96] propuseram e implementaram no Condor o *condor flocking*, que permite a criação de acordos entre instituições para compartilhar seus recursos. Para criar um grid usando o fbcking, as instituições negociam as permissões e prioridades a ser dadas a cada uma delas. Sempre que uma nova instituição entra no grid, uma nova negociação acontece. A necessidade desta negociação limita a capacidade do fbcking de lidar com situações de uso em produção, onde o ambiente de compartilhamento é altamente dinâmico [TTL03]. Como veremos na Seção 3.2.2, um outro mecanismo foi proposto para substituir o fbcking.

O Computational Co-op [CM99] propôs mecanismos para agregar sites para formar um grid usando a metáfora de uma cooperativa. Esses mecanismos permitem a todos os sites controlar quanto de seus recursos está sendo utilizado pelo grid e provê garantias de quanto dos recursos do grid cada site será capaz de usar. Isso é feito usando um escalonador de divisão proporcional dos recursos compartilhados baseado em fi chas. As fi chas são usadas pelos usuários para acessar os recursos locais e do grid, obtendo prioridade de acordo com a quantidade de fi chas gasta. Entretanto, a necessidade de negociações entre os donos dos sites para defi nir a divisão das fi chas do grid e a impossibilidade da transferência de fi chas ou seu consumo fazem com que o Co-op não seja fexível o sufi ciente para ambientes tão dinâmicos quanto grids.

O objetivo central deste trabalho é criar um mecanismo que facilite o compartilhamento de recursos computacionais entre diferentes instituições incentivando a contribuição de recursos para este compartilhamento. Esperamos com isso contribuir para o estado da arte em composição de grids.

3.2 Composição de Grids Computacionais

Nesta seção, comparamos nossos resultados aos principais sistemas de compartilhamento de recursos computacionais que visam automatizar a composição de grids computacionais.

Identificamos duas abordagens principais para esses sistemas: economia grid e compartilhamento par-a-par. Essas duas abordagens não representam uma classificação estrita, o único propósito deste agrupamento é facilitar a discussão apresentada neste texto.

3.2.1 Economia Grid

Categorizamos como economia grid os esforços que visam criar economias de recursos computacionais baseadas em moedas, usando o modelo de mercado para criar incentivos monetários à contribuição. O *Compute Power Market* [BV00] é um projeto que objetiva criar uma comunidade par-a-par onde um usuário pode pagar para obter acesso a recursos compartilhados usando, por exemplo, seu cartão de crédito. O Nimrod/G [ABG02] é um sistema para a execução de aplicações de varredura de parâmetros que implementa os conceitos de uma arquitetura grid para a economia computacional (GRACE) [BAG00]. Esses conceitos permitem que um cliente grid negocie o acesso a recursos disponibilizados por provedores e pague a estes pelos serviços obtidos. Um ponto importante a notar é que para permitir a negociação entre consumidores e provedores de serviço usando uma moeda global como proposto no Nimrod/G e no Compute Power Market, uma infra-estrutura para a negociação, pagamentos e auditoria seguras deve ser implantado. Atribuímos à necessidade desta infra-estrutura a inexistência de sistemas baseados em economia grid em produção hoje (que sejam de nosso conhecimento). O OurGrid e a Rede de Favores objetivam resolver o problema da composição de grids apenas para aplicações BoT. Argumentamos que é possível fazer isso sem a criação de uma economia baseada em moeda. Por conseguinte, a solução que propusemos não depende de infra-estruturas não disponíveis hoje, como as propostas baseadas na economia grid.

3.2.2 Compartilhamento Par-a-Par

Butt et al. [BZH03] propõem a criação de um mecanismo de compartilhamento par-a-par entre instituições usando o Condor. Este o mecanismo par-a-par visa substituir o fbcking descrito na Seção 3.1. Para isso, ele facilita a criação de uma comunidade onde nós podem facilmente entrar e sair. Os projetos Triana [TSWP03] e JNGI [VNRS02] também visam facilitar a criação de grids considerando-os redes de compartilhamento e recursos par-a-par.

Utilizando a abordagem par-a-par, os três se propõem a ter implantação simples. A questão do incentivo à colaboração, entretanto, não é atacada em nenhum destes trabalhos. Uma comunidade formada utilizando apenas estes mecanismos se baseia apenas no altruísmo de seus participantes para ter recursos disponíveis.

Um outro projeto que pode ser considerado par-a-par por usar os recursos de todos os participantes do sistema é o BOINC [PKA+03]. Este projeto é o sucessor de uso geral do SETI@home [ACK+02], e permite a um cliente utilizar os recursos ociosos de um conjunto de contribuintes. Esses contribuintes são motivados a doar recursos por quererem ajudar à causa da aplicação que será executada no grid. A necessidade deste tipo de motivação restringe a utilidade da infra-estrutura bem como a motivação de instituições a colaborarem com grandes quantidades de recursos. Note que nem todos os pares que contribuem para o sistema podem submeter suas aplicações para executar no grid.

Uma abordagem similar à nossa é a proposta em Chun et al. [CFV03]. Como nós, os autores creditam a inexistência de soluções de gerenciamento de recursos em sistemas distribuídos em larga escala amplamente adotadas devido à complexidade inicial das propostas. Eles sugerem a construção de uma economia básica baseada em escambo, visando a utilização de mecanismos mais complexos sobre esta economia básica apenas após sua adoção inicial. A proposta da criação desta economia é construída sobre uma arquitetura para o compartilhamento de recursos de forma segura chamada SHARP (*secure highly available resource peering*). A SHARP [FCC+03] se baseia na troca de fichas entre seus participantes, de forma semelhante ao Computational Co-op. Porém, na SHARP as fichas são apenas indicativos, e não garantias de acesso a recursos por um determinado período. Além disso, é possível repassar subconjuntos dos recursos indicados nas fichas para outros pares. Essa possibilidade, juntamente com a possibilidade de criar um número de fichas arbitrário, dependendo dos níveis de utilização dos recursos possuídos por cada par parecem prover a flexibilidade ausente no Computational Co-op. Por fim, uma versão da SHARP foi implantada em um cenário real e tem resultados promissores.

O trabalho de Chun et al. difere do nosso principalmente ao enfocar a segurança na delegação do acesso aos recursos da comunidade, e não o incentivo à colaboração. Para garantir essa segurança, um par só consegue fichas de pares com quem ele tem uma relação de confiança direta ou indireta (através da confiança de algum par no qual ele tem confiança).

Para negociar com um par com quem o par consumidor não tem uma relação de confiança, é preciso criar uma cadeia de negociações. Nossa abordagem é diferente ao enfocar a intenção de facilitar o acesso a todos os recursos da comunidade. Não é possível delegar o direito ao acesso a recursos, mas é possível acessar diretamente os recursos da comunidade inteira. Na Rede de Favores não há negociações de recursos, e atacamos diretamente o problema do par que não colabora com a comunidade, marginalizando-o.

3.3 Esquemas de Reputação Par-a-Par

O resultado principal desta dissertação é a possibilidade do incentivo à cooperação no compartilhamento de recursos para rodar aplicações BoT usando um esquema de reputação autônomo. Nosso esquema, a Rede de Favores, difere das propostas no estado da arte em esquemas de reputação em sistemas par-a-par por ser descentralizado e inteiramente autônomo.

Em esquemas de reputação *centralizados*, como o utilizado no eBay [eBa03], informação sobre a reputação dos pares é mantida em um banco de dados centralizado, acessível a todos os pares. Este banco centralizado é um ponto único de falha, e para mantê-lo disponível é necessária uma infra-estrutura de tolerância à falhas. Além disso, para garantir que ele não seja um gargalo no desempenho do sistema, é necessário também garantir sua escalabilidade. A necessidade dessa infra-estrutura para garantir o serviço de um componente central no sistema contrasta com o nosso objetivo de obter uma solução que dependa de um mínimo de infra-estruturas para sua implantação.

Contudo, a intenção de não depender de componentes centrais é comum a diversos sistemas par-a-par. Existem diversas propostas de esquemas de reputação *decentralizados* para contornar o uso desses componentes. Nesse tipo de esquema, a informação sobre a reputação dos pares está disponível para todos, mas distribuída em diferentes partes do sistema. Por exemplo, no esquema P2PRep [DdVPS03], cada par armazena informação sobre suas interações com outros pares, e no EigenRep [KSGM03] além desta informação, alguns pares armazenam valores globais derivados de múltiplos valores locais sobre a reputação de outros pares. Em esquemas de reputação distribuídos, um par obtém informação do sistema a respeito de outro par usando um protocolo de obtenção.

Uma questão com a qual todos os protocolos de obtenção devem lidar é garantir que a

informação obtida sobre um par é confiável, pois pares maliciosos podem fraudar a informação que eles armazenam. Para garantir essa confiança, P2PRep utiliza votação para obter a informação sobre um par, heurísticas para descobrir grupos de pares maliciosos votando juntos e uma estrutura criptográfica para verificar as identidades dos pares envolvidos em uma transação. Diferentemente, no EigenRep alguns "pares mãe" computam e armazenam o valor global de reputação de um par. Os pares mãe acham os pares cujas reputações eles devem manter e são achados pelos que desejam essa informação através de uma tabela hash distribuída.

Com a Rede de Favores mostramos que, para o cenário do compartilhamento de recursos para a execução de aplicações BoT, é possível incentivar a colaboração com a comunidade sem a obtenção de um valor global de reputação para um par. Em lugar disso, é possível para os pares usar apenas informação local e totalmente confiável para priorizar seus consumidores e ainda assim promover a colaboração no sistema como um todo. O uso de informação local apenas torna desnecessária qualquer infra-estrutura de extração de valores globais de reputação ou de garantia na confiança desses valores, tornando mais simples a implantação de um sistema que use a Rede de Favores.

Dois outros sistemas par-a-par têm relação com o uso de reputação autônoma: GNUnet [Gro03] e BitTorrent [Coh03]. GNUnet é uma rede de compartilhamento anônimo de arquivos que usa uma economia baseada em confiança para proteger a rede de ataques de *denial of service*. Na GNUnet, a fim de manterem-se anônimos, os pares fazem requisições para seus vizinhos apenas, e estes repassam as requisições como se fossem deles. Por conseguinte, as respostas para uma requisição feita por um par só são retornadas por seus vizinhos. Agindo desta forma, os pares são capazes de identificar quais dos seus vizinhos retornam requisições, e ganham confiança neles. Em um momento de contenção de recursos, esses pares são priorizados. Apesar de focar na defesa de ataques de pares maliciosos, esse mecanismo também incentiva a cooperação dos pares corretos. A principal diferença entre essa proposta e a nossa é a consideração na forma de comunicação nos pares. Consideramos na Rede de Favores que um par pode se comunicar com qualquer outro par da comunidade (e potencialmente com todos), e não apenas com seus vizinhos. Por outro ângulo, podemos argumentar que na GNUnet os pares podem apenas consumir os recursos de seus vizinhos, sendo ávidos nesse escopo. Assim, os dois esquemas de reputação tornam-se bastante próximos.

BitTorrent é um sistema de compartilhamento de largura de banda entre pares que estão baixando um mesmo arquivo de um servidor. Os pares baixam trechos do arquivo uns dos outros sempre que possível, aliviando ou mesmo tornando desnecessário após algum tempo o servidor original. Para decidir a que outros pares doar largura de banda quando há contenção, os pares de uma rede BitTorrent usam um mecanismo de *Tit-for-Tat*, colaborando com aqueles que estão colaborando. Como os pares não fi cam muito tempo em uma rede BitTorrent, não há a necessidade de uma memória sobre a colaboração de outros pares.

Apesar da existência desses mecanismos de reputação autônoma, não há, em nosso conhecimento, nenhuma análise do desempenho ou comprovação experimental do funcionamento de nenhum deles. Com nosso trabalho, mostramos para o caso do compartilhamento de recursos computacionais para a execução de aplicações BoT que um esquema similar ao usado nesses sistemas incentiva à colaboração.

Capítulo 4

Guia para os Apêndices

Este capítulo apresenta um guia para a leitura dos apêndices da dissertação. Cada um dos apêndices é um artigo científi co cujo primeiro autor é o autor desta dissertação. Esses apêndices completam o trabalho apresentado nesta dissertação, detalhando aspectos apresentados brevemente nela. A Seção 4.1 detalha o projeto da Comunidade OurGrid. Na Seção 4.2, apresentamos uma investigação mais detalhada sobre a função usada para calcular a reputação de um par na Rede de Favores. Por fi m, a Seção 4.3 contém um estudo das condições necessárias para que a Rede de Favores incentive à colaboração quando os consumidores que compõem o sistema não são ávidos.

4.1 Concepção do OurGrid

O Apêndice A ilustra uma primeira visão do problema, motiva nossa abordagem e contém a concepção de nossa solução. A comunidade OurGrid é o centro do trabalho. A Rede de Favores ainda em estágio inicial é identifi cada como mecanismo de incentivo. Contudo, as investigações são ainda iniciais.

Alguns pontos no modelo usado nesse artigo diferem dos estudos posteriores. Em especial, nesse artigo não consideramos free-riders. Portanto, consideramos que os pares do sistema têm apenas dois estados: consumidor ou provedor. Negligenciamos o estado de ociosidade. A investigação conduzida nesse apêndice visa verifi car apenas eqüidade na divisão dos recursos disponíveis no sistema.

Um segundo ponto merece uma ressalva ainda na modelagem dos pares: nas simula-

ções desse artigo, cada site é composto de uma quantidade de recursos e de uma demanda dadas por distribuições independentes. Apesar de, mesmo com esta independência, termos verifi cado a eqüidade no sistema através da razão de favores (*Favor Ratio*, no apêndice), acreditamos que a outra métrica utilizada no artigo produz observações imprecisas. Nossa segunda métrica, chamada *Resource Gain* (RG), é a razão entre a quantidade de recursos total obtida por um par e a quantidade de recursos locais que ele consumiu. Desta forma, o valor de RG é função da quantidade de recursos que um par possui e também de sua demanda. O problema é que a análise apresentada no artigo mostra a evolução de RG no tempo apenas para diferentes demandas de diferentes pares, sem considerar a quantidade de recursos possuída por cada um deles. Em estudos posteriores, a métrica RG foi deixada de lado.

Finalmente, o artigo apresenta o projeto de alto nível de um par OurGrid. Esse par está sendo implementado. Em Janeiro de 2004, o software se encontra em uma versão alfa. Naturalmente, durante o desenvolvimento desta versão, modifi cações no projeto apresentado no apêndice foram feitas. A mais signifi cativa foi a decisão da substituição do modelo de envio de tarefa para o provedor. No protocolo descrito no Apêndice A, quando usando um recurso estrangeiro, o cliente manda sua tarefa para que o provedor a execute em um de seus processadores. Na implementação do par OurGrid, optamos por dar o acesso direto ao processador para o cliente. Assim, em lugar de enviar os arquivos de entrada e a tarefa para o par provedor, o cliente pode fazer quaisquer atividades com o processador durante o tempo em que este está alocado para ele. Acreditamos que esse modelo facilita (i) a detecção de falhas no processador por parte do cliente, uma vez que ele pode verifi car se o processador ainda responde periodicamente; e (ii) a execução de atividades como transferência de dados e execução de uma tarefa em paralelo, permitindo que o cliente, por exemplo, transfi ra os dados de entrada de uma próxima tarefa durante a execução da tarefa atual.

4.2 A Rede de Favores para Consumidores Ávidos

A evolução do trabalho apresentado no Apêndice A foi a investigação em detalhes da Rede de Favores como mecanismo de incentivo à colaboração. O Apêndice B traz esta investigação, agora considerando free-riders, para cenários onde os consumidores são ávidos. Consumi-

dores ávidos são aqueles capazes se benefi ciar consumindo quaisquer recursos disponíveis. Como discutido no Capítulo 2, acreditamos que esta consideração é realista para o cenário das aplicações BoT em grids.

Alguns resultados apresentados no Apêndice B são semelhantes aos contidos no Capítulo 2 da dissertação. Esse apêndice contém uma análise analítica que mostra, dado o modelo do sistema, as condições necessárias para que um esquema de reputação incentive a colaboração. Com base nesta análise, são apresentados resultados de simulações que mostram que a Rede de Favores satisfaz as condições estabelecidas pela análise um conjunto de cenários signifi cativo.

Além desses resultados, esse trabalho contém considerações sobre duas extensões na função usada para o cálculo da reputação local de um par. A primeira extensão é o uso de reputação não-negativa, que visa tornar a Rede de Favores robusta a um comportamento malicioso relacionado a esquemas de reputação: a troca de identidade. Tipicamente, é fácil trocar de identidade em uma rede par-a-par. Em muitos casos, basta reinstalar o software. Ao trocar de identidade e voltar como um novato, um par "zera" sua reputação na comunidade. Um par mal-intencionado pode lucrar apenas consumindo e mudando de identidade periodicamente, se a comunidade não tiver uma forma de prevenir isso.

A segunda extensão é o uso de uma função que também valorize o conhecimento prévio de um par para priorizá-lo. Esta função visa investigar o uso de uma valoração mais próxima do comportamento humano para as reputações no sistema. Os resultados apresentados indicam que os pares do sistema se agrupam em "círculos sociais". A investigação das conseqüências desse agrupamento é um dos trabalhos futuros desta dissertação.

4.3 Condições para que a Rede de Favores Incentive a Colaboração

Após a discussão da motivação para a Rede de Favores e a análise de seu desempenho para o cenário específi co dos consumidores ávidos, feitos nos dois apêndices anteriores, o Apêndice C apresenta uma investigação mais geral das condições necessárias para que a Rede de Favores seja efi caz em incentivar a colaboração. Especifi camente, esse trabalho deixa de lado a consideração de que todos os consumidores são ávidos.

Baseado na Rede de Favores, consideramos o uso de reputação apenas para priorizar os colaboradores na doação de recursos. Assim, um par nunca se recusa a doar um recurso se este não é contestado. No Apêndice C, uma avaliação analítica mostra que, desta forma, mesmo um esquema de reputação perfeito (que sempre prioriza os colaboradores) não incentiva a colaboração em qualquer cenário. Definidas as condições para que um esquema de reputação perfeito incentive a colaboração, simulações comparam o desempenho de um esquema perfeito e da Rede de Favores. Os resultados mostram que na maior parte dos cenários investigados, após a Rede de Favores obter informação sobre a comunidade, o desempenho dos dois esquemas de reputação é similar.

Capítulo 5

Conclusões e Trabalhos Futuros

Este capítulo contém as considerações finais sobre a dissertação. Na Seção 5.1 apresentamos um apanhado do significado de nossos resultados. Em seguida, discutimos sugestões de trabalhos subseqüentes aos resultados apresentados nesta dissertação na Seção 5.2.

5.1 Conclusões

Esta dissertação mostra que é possível incentivar a colaboração no compartilhamento de recursos computacionais para a execução de aplicações Bag-of-Tasks usando um esquema de reputação autônomo que desenvolvemos, chamado de Rede de Favores. Este esquema incentiva à colaboração apenas através do comportamento autônomo dos componentes do sistema. Usando apenas esse comportamento, a Rede de Favores possui implementação e implantação simples. Isso viabiliza a implantação de soluções que a utilizem em produção em curto prazo. Acreditamos que a criação desse tipo de solução e sua utilização em produção serão de grande utilidade para diversos grupos de pesquisa que têm aplicações BoT e necessitam de muitos recursos computacionais.

Para atender a esses grupos de pesquisa, estamos implementando a comunidade OurGrid. O estudo da Rede de Favores nos indica que é possível a utilização de uma solução baseada na simplicidade para incentivar a colaboração nessa comunidade. Esperamos com a implementação obter uma prova de conceito desse resultado. Por fim, com o OurGrid usando a Rede de Favores, esperamos compor um grid do qual seja possível colher dados sobre a utilização de grids em produção. Com esses dados será possível projetar a evolução do OurGrid

com base em experiências de uso. Assim esperamos poder empreender nossos esforços na solução de questões mais próximas dos usuários atuais da computação em grid.

5.2 Trabalhos Futuros

Identifi camos três sugestões de trabalhos futuros como evoluções dos resultados apresentados nesta dissertação.

5.2.1 Concepção de um mecanismo robusto de contabilização dos favores

Nosso estudo da Rede de Favores, assim como sua atual implementação no OurGrid, considera que não há pares mal-intencionados na contabilização dos favores. Consideramos que o provedor sempre retorna o valor correto do favor prestado para o consumidor. Embora uma primeira instância da comunidade OurGrid seja possível com essa consideração, ela precisa tornar-se desnecessária com a evolução do sistema.

Um par pode se benefi ciar reportando valores maiores que os reais para seus favores, se a comunidade não for capaz de perceber esse comportamento. Como discutido na Seção 2.5, uma possibilidade identifi cada para tornar a comunidade robusta a esse tipo de comportamento é a extensão do mecanismo de tolerância à sabotagem proposto por Sarmenta [Sar02]. Este mecanismo se baseia na replicação de tarefas para ganhar credibilidade nos resultados reportados pelos provedores de serviço. Acreditamos que o uso dessa replicação também para a verifi cação da contabilidade reportada pode dar aos consumidores a capacidade de detectar provedores mal-intencionados.

5.2.2 Consideração de uma comunidade que compartilha diversos bens

No modelo usado nesta dissertação, consideramos que todos os bens da comunidade são intercambiáveis. Acreditamos que essa consideração é razoável para o cenário da implantação do OurGrid. Nossa experiência sugere que o recurso mais escasso para aplicações BoT é CPU, e que um usuário freqüentemente pode recompilar sua aplicação para as plataformas mais comuns. Todavia, não sabemos se um número signifi cativo de aplicações usará recur-

sos que são consideravelmente mais escassos na comunidade. Só teremos essa informação, e, portanto, a confirmação da necessidade de abandonar nossa consideração, após o uso da Rede de Favores no cenário específico do OurGrid.

De qualquer forma, para tornar os resultados da Rede de Favores mais gerais — e, portanto, aplicáveis a mais cenários e sistemas —, é importante considerar o caso em que uma comunidade compartilha bens não intercambiáveis. Os pares podem possuir demandas diferentes para os diversos bens, e, assim, criar bens mais valiosos. A existência de bens de diferentes valores cria novos problemas para a contabilização dos favores na comunidade. Torna-se necessário definir fatores de conversão entre os valores desses bens, e, potencialmente, certificar-se de que esses valores de conversão são globais.

5.2.3 Investigação do uso de funções de reputação mais elaboradas

A função que usamos para o cálculo da reputação local de um par nos resultados apresentados nesta dissertação foi a mais simples possível. Definimos a reputação local de um par B em A como o saldo entre os favores recebidos de B por A e os prestados por A para B. Outras funções podem ser usadas para estender o comportamento obtido através do uso dessa.

Investigamos inicialmente dois aprimoramentos no Apêndice B: o uso de reputação não-negativa e o uso de uma função com um termo sub-linear. O primeiro aprimoramento visa tornar a comunidade robusta à tentativa de pares maliciosos de se beneficiar mudando de identidade freqüentemente. e tem o comportamento emergente da comunidade quando a utilizando bem entendido. O uso da função com termo sub-linear, entretanto, não foi completamente investigado ainda. Essa função visa aproximar mais o comportamento da reputação na Rede de Favores com o que acreditamos ser a forma como seres humanos valoram suas reputações. Ela melhora um pouco o desempenho da identificação de free-riders em comparação com o uso da reputação não-negativa sem o termo sub-linear, porém o uso desta função na comunidade parece criar vínculos “sociais” entre os pares. Aparentemente, com o tempo e termo sub-linear da função cresce e os pares passam a doar recursos não só com base nas doações que receberam, mas também influenciados pelo tempo com que eles vêm trocando favores com os outros pares.

Considerando uma rede onde os nós são os pares do sistema e há conexões entre pares que trocam favores freqüentemente, suspeitamos que esse agrupamento pode facilitar a detecção

de pares mal-intencionados na comunidade. A estratégia de sabotagem mais difícil de ser detectada pela comunidade é a em que o par mal-intencionado sabota um pouco muitos pares. Ao sabotar muito um mesmo par, o par mal-intencionado se expõe e facilita sua descoberta. Caso a topologia da rede mencionada mostre que os nós se agrupam em pequenos círculos, os nós provavelmente interagirão muito freqüentemente com um número pequeno de outros nós. Se os vínculos em um círculo forem fortes, pode ser difícil mudar de círculo social. Para entrar em um círculo, seria preciso primeiro doar uma quantidade de recursos signifi cativa para seus membros. Isso difi cultaria a sabotagem de muitos pares, obrigando os sabotadores a doarem uma quantidade signifi cativa de recursos para um mesmo par para lucrar com a sabotagem e, por conseguinte, a se exporem mais. Por outro lado, pode ser que a necessidade de doar uma quantidade de recursos signifi cativa para obter recursos de volta da comunidade diminua o incentivo à entrada de novatos na comunidade. O impacto do agrupamento dos pares nesse aspecto da comunidade também precisa ser estudado.

Bibliografia

[ABG02] David Abramson, Rajkumar Buyya, and Jonathan Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems (FGCS) Journal*, 18:1061–1074, 2002.

[ACBR03] Nazareno Andrade, Walfredo Cirne, Francisco Brasileiro, and Paulo Roisenberg. OurGrid: An approach to easily assemble grids with equitable resource sharing. In *Anais do 9o Workshop on Job Scheduling Strategies for Parallel Processing*, Junho 2003.

[ACK+02] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[AGK00] D. Abramson, J. Giddy, and L. Kotler. High performance parametric modeling with Nimrod/G: Killer application for the global grid? In *Anais do IPDPS'2000*, pages 520–528. IEEE CS Press, 2000.

[AH00] Eytan Adar and Bernardo Huberman. Free riding on gnutella. *First Monday*, 5(10), 2000. http://www.firstmonday.dk/.

[BAG00] R. Buyya, D. Abramson, and J. Giddy. An economy driven resource management architecture for computational power grids. In *International Conference on Parallel and Distributed Processing Techniques and Applications*, 2000. http://citeseer.nj.nec.com/299926.html.

[BB03] Alexander Barmouta and Rajkumar Buyya. GridBank: A Grid Accounting Ser-

vices Architecture (GASA) for distributed systems sharing and integration. In *Anais do 26th Australasian Computer Science Conference (ACSC2003)*, 2003.

[BM93] Ozalp Babaoglu and Keith Marzullo. *Distributed Systems*, chapter 4: Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms. Addison-Wesley, 1993.

[BV00] Rajkumar Buyya and Sudharshan Vazhkudai. Compute Power Market: Towards a Market-Oriented Grid. In *The First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, Beijing, China, 2000. IEEE Computer Society Press.

[BWF+96] Fran Berman, Richard Wolski, Silvia Figueira, Jennifer Schopf, and Gary Shao. Application-level scheduling on distributed heterogeneous networks. In *Anais do Supercomputing'96*, 1996.

[BZH03] Ali Raza Butt, Rongmei Zhang, and Y. Charlie Hu. A self-organizing fbck of condors. In *Anais do Supercomputing 2003*, Novembro 2003.

[CBS+03] Walfredo Cirne, Francisco Brasileiro, Jacques Sauvé, Nazareno Andrade, Daniel Paranhos, Elizeu Santos-Neto, Raissa Medeiros, and Fabrício Silva. Grid computing for Bag-of-Tasks applications. In *Anais do IFIP I3E2003*, Setembro 2003.

[CFK+98] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *Anais do IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, 1998.

[CFV03] Brent N. Chun, Yun Fu, and Amin Vahdat. Bootstrapping a distributed computational economy with peer-to-peer bartering. In *Anais do 1st Workshop on Economics of Peer-to-Peer Systems*, Junho 2003.

[CHY02] H. Casanova, J. Hayes, and Y. Yang. Algorithms and software to schedule and deploy independent tasks in grids environments. In *Anais do Workshop on Distributed Computing, Metacomputing and Resource Globalization*, 2002.

[CLZB00] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for scheduling parameter sweep applications in grid environments. In *Anais do 9th Heterogeneous Computing Workshop*, pages 349–363, Cancun, Mexico, Maio 2000. IEEE Computer Society Press.

[CM99] W. Cirne and K. Marzullo. The computational Co-op: Gathering clusters into a metacomputer. In *Anais do PPS/SPDP'99 Symposium*, 1999.

[CM01] Walfredo Cirne and Keith Marzullo. Open Grid: A user-centric approach for grid computing. In *Anais do 13th Symposium on Computer Architecture and High Performance Computing*, 2001.

[COBw00] Henri Casanova, Graziano Obertelli, Francine Berman, and Rich wolski. The apples parameter sweep template: User-level middleware for the grid. In *Supercomputing Conference (SC'2000)*, 2000.

[Coh03] Bram Cohen. Incentives build robustness in BitTorrent. In *Anais do 1st Workshop on Economics of Peer-to-Peer Systems*, Junho 2003.

[CPC+03] Walfredo Cirne, Daniel Paranhos, Lauro Costa, Elizeu Santos-Neto, Franscisco Brasileiro, Jacques Sauvé, Fabrício Alves Barbosa da Silva, and Cirano Silveira. Running bag-of-tasks applications on computational grids: The MyGrid approach. In *Anais do ICCP'2003 - International Conference on Parallel Processing*, Outubro 2003. (to appear).

[DdVPS03] E. Damiani, S. De Capitani di Vimercati, S. Parabosci, and P. Samaranti. Managing and sharing servents' reputations in P2P systems. *IEEE Transactions on Data and Knowledge Engineering*, 15(4):840–854, Julho/Agosto 2003. http://seclab.dti.unimi.it/p2prep/.

[Del03] Chrysanthos Dellarocas. Effi ciency and robustness of binary feedback mechanisms in trading environments with moral hazard. Working Paper 4297-03, MIT - Sloan School of Management, Janeiro 2003.

[eBa03] eBay. Reputation - eBay feedback: Overview, 1995-2003. http://pages.ebay.com/help/confi dence/reputation-ov.html.

[ELvD+96] D.H.J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A worldwide fbck of Condors: Load sharing among workstation clusters. *Future Generation Computer Systems*, 12:53–65, 1996.

[eMu03] eMule-Project.net. eMule homepage. http://www.emule-project.net/, 2003.

[FCC+03] Yun Fu, Jeff Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. SHARP: An architecture for secure resource peering. In *Anais do 19th ACM Symposium on Operating System Principles*, Outubro 2003.

[FK97] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[FK98] I. Foster and C. Kesselman. The Globus project: A status report. In *Anais do IPPS/SPDP '98 Heterogeneous Computing Workshop*, pages 4–18, 1998.

[FKN+02] I. Foster, C. Kesselman, J. Nick, S. Tuecke, Open Grid Service Infrastructure WG, and Global Grid Forum. The Physiology of the Grid: An Open Grid Services Architecture for distributed systems integration, 2002.

[Fos01] Ian Foster. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150, 2001.

[FTF+02] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steve Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5:237–246, 2002.

[GM01] Philippe Golle and Ilya Mironov. Uncheatable distributed computations. *Lecture Notes in Computer Science*, 2020:425–441, 2001.

[Gri03] Grid Economic Services Architecture Working Group in the Global Grid Forum. GESA homepage, 2002-2003. http://www.doc.ic.ac.uk/~sjn5/GGF/gesa-wg.html.

[Gro03] Christian Grothoff. An Excess-Based Economic Model for Resource Allocation in Peer-to-Peer Networks. *Wirtschaftsinformatik*, June 2003.

[HM98] Fred Howell and Ross McNab. SimJava: a discrete event simulation package for Java with applications in computer systems modelling. In *Anais do First International Conference on Web-based modelling and simulation*. Society for Computer Simulation, 1998.

[JXT04] JXTA.org. Project JXTA homepage, 2004.

[KSGM03] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. EigenRep: Reputation management in P2P networks. In *Twelth International World Wide Web Conference*, Budapeste, Hungria, Maio 2003.

[KW03] H. T. Kung and Chun-Hsin Wu. Differentiated admission for peer-to-peer systems: incentivizing peers to contribute their resources. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Junho 2003.

[LFSC03] Kevin Lai, Michal Feldman, Ion Stoica, and John Chuang. Incentives for cooperation in peer-to-peer networks. In *Anais do Workshop on Economics of Peer-to-Peer Systems*, Junho 2003.

[LLM88] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor - a hunter of idle workstations. In *Anais do 8th International Conference of Distributed Computing Systems*, 1988.

[LRW03] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the KaZaA network. In *Anais do 3rd IEEE Workshop on Internet Applications*, Junho 2003.

[OLK03] B. C. Ooi, C.Y. Liau, and K.L.Tan. Managing trust in peer-to-peer systems using reputation-based techniques. In *International Conference on Web Age Information Management (WAIM'03)*, Agosto 2003.

[our03] ourgrid.org. OurGrid project site, 2003.

[PCB03] Daniel Paranhos, Walfredo Cirne, and Francisco Brasileiro. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In *Anais da Euro-Par 2003: International Conference on Parallel and Distributed Computing*, 2003.

[PKA+03] Andrew Page, Thomas Keane, Richard Allen, Thomas J. Naughton, and John Waldron. Multi-tiered distributed computing platform. In *Anais da 2nd International Conference on the Principles and Practice of Programming in Java*, Junho 2003.

[RF02] Matei Ripeanu and Ian Foster. Mapping the Gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In *First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[RRSF03] Kavitha Ranganathan, Matei Ripeanu, Ankur Sarin, and Ian Foster. To share or not to share: An analysis of incentives to contribute in collaborative fi le sharing environments. In *Anais do Workshop on Economics of Peer-to-Peer Systems*, Junho 2003.

[RZFK00] Paul Resnick, Richard Zeckhauser, Eric Friedman, and Ko Kuwabara. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[Sar02] Luis F. G. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, 2002.

[SBSS98] J. R. Stiles, T. M. Bartol, E. E. Salpeter, and M. M. Salpeter. Monte carlo simulation of neuromuscular transmitter release using MCell a general simulator of cellular physiological processes. *Computational Neuroscience*, pages 279–284, 1998.

[SCF+00] Shava Smallen, Walfredo Cirne, Jaime Frey, Francine Berman, Rich Wolski, Mei-Hui Su, Carl Kesselman, Steve Young, and Mark Ellisman. Combining workstations and supercomputers to support grid applications: The parallel tomography experience. In *Anais do HCW'2000 - Heterogeneous Computing Workshop*, 2000.

[SGG02] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer fi le sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[SNTF+01] E. L. Santos-Neto, L. E. F. Tenório, E. J. S. Fonseca, S. B. Cavalcanti, and J. M. Hickmann. Parallel visualization of the optical pulse through a doped optical fi ber. In *Anais do Annual Meeting of the Division of Computational Physics*, Boston, MA, USA, 2001.

[SS96] J. Smith and S. K. Shrivastava. A system for fault-tolerant execution of data and compute intensive programs over a network of workstations. In *Lecture Notes in Computer Science*, volume 1123. IEEE Press, 1996.

[TSWP03] Ian Taylor, Matthew Shields, Ian Wang, and Roger Philp. Distributed p2p computing within triana: A galaxy visualization test case. In *Anais do International Parallel and Distributed Processing Symposium 2003 (IPDPS'03)*, Abril 2003.

[TTL03] Douglas Thain, Todd Tannenbaum, and Miron Livny. *Grid Computing: Making the Global Infrastructure a Reality*, chapter 11 - Condor and the grid. John Wiley, 2003.

[VNRS02] Jerome Verbeke, Neelakanth Nadgir, Greg Ruetsch, and Ilya Sharapov. Framework for peer-to-peer distributed computing in a heterogeneous, decentralized environment. In *GRID 2002*, volume 2536 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2002.

[WPBB01] R. Wolski, J. Plank, J. Brevik, and T. Bryan. Analyzing market-based resource allocation strategies for the computational grid. *International Journal of High-performance Computing Applications*, 15(3), 2001.

[YM02] T. Yamagishi and M. Matsuda. Improving the lemons market with a reputation system: An experimental study of internet auctioning, May 2002. http://joi.ito.com/archives/papers/Yamaghishi_ASQ1.pdf.

Appendix A

OurGrid: An approach to easily assemble grids with equitable resource sharing

Nazareno Andrade[1], Walfredo Cirne[1], Francisco Brasileiro[1], Paulo Roisenberg[2]

[1]Universidade Federal de Campina Grande

[2]Hewllet Packard Brazil

Abstract: Available grid technologies like the Globus Toolkit make possible for one to run a parallel application on resources distributed across several administrative domains. Most grid computing users, however, don't have access to more than a handful of resources onto which they can use this technologies. This happens mainly because gaining access to resources still depends on personal negotiations between the user and each resource owner. To address this problem, we are developing the OurGrid resources sharing system, a peer-to-peer network of sites that share resources equitably in order to form a grid to which they all have access. The resources are shared accordingly to a *network of favors* model, in which each peer prioritizes those who have credit in their past history of bilateral interactions. The emergent behavior in the system is that peers that contribute more to the community are prioritized when they request resources. We expect, with OurGrid, to solve the access gaining problem for users of *bag-of-tasks* applications (those parallel applications whose tasks are

independent).[1]

A.1 Introduction

To use grid computing, a user must assemble a grid. A user must not only have the technologies to use grid computing, but also, she must have access to resources on which she can use these technologies. For example, to use resources through the Globus Toolkit [FK98], she must have access, i.e., permission to use resources on which Globus is installed.

Today, the access gaining to grid resources is done via personal requests from the user to each resource owner. To run her application on the workstations of some laboratories in a university, a user must convince the system administrator of each laboratory to give her access to their system's workstations. When the resources the user wishes to use cross institutional boundaries, the situation gets more complicated, as possibly different institutional policies come in. Thus, it is very difficult for one to gain access to more than a handful of resources onto which one can use grid computing technologies to run one's applications.

As resource owners must provide access to their resources to allow users to form grids, there must be some interest in providing resources to grid users. Also, as several grid users may demand the same resource simultaneously, there must be mechanisms for dealing with conflicting requests for resources, arbitrating them. Since these problems can be seen as offer and demand problems, approaches to solve them have been based on *grid economy* [ABG02; BV00; WPBB01; BB03], which means using, on grids, economic models from real markets.

Although models that mimic real markets have the mechanisms to solve the problems of a grid's offer and demand, they rely on a yet-not-available infrastructure of electronic monetary transactions. To make possible for users to securely verify what they have consumed and pay for it, there must be mature and well deployed technologies for electronic currency and banking. As these technologies are not widely deployed yet, the actual use of the economic mechanisms and architectures in real settings is postponed until the technologies are mature and the infrastructure needed to use them is available.

[1] Originalmente publicado nos anais do 9th Workshop on Job Scheduling Strategies for Parallel Processing, em Junho de 2003. / Originally published in the proceeding of the 9th Workshop on Job Scheduling Strategies for Parallel Processing, in June of 2003.

Nevertheless, presently there exists demand for grids to be used in production. Aiming to provide, in short term, an infrastructure that addresses this demand for an expressive set of users, we are developing OurGrid. The OurGrid design is based on a model of resource sharing that provides equity with a minimum of guaranties needed. With it, we aim to provide an open, extensible and easy to deploy platform, suitable for running a useful set of grid applications for users willing to share their resources in order to obtain access to the grid.

The type of application for which OurGrid intends to provide resources to are those loosely coupled parallel applications known as *bag-of-tasks* (BoT) applications [CBS+03; SS96]. BoT are those parallel applications composed of a set of independent tasks that need no communication among them during execution. Many applications in areas such as computational biology [SBSS98], simulations, parameter sweep [AGK00] and computer imaging [SCF+00; SNTF+01] fit into this definition and are useful to large communities of users.

Additionally, from the research perspective, there exists demand for understanding grid usage requirements and patterns in real settings. With a system such as OurGrid in production for real users, we will be able to gather valuable information about the needs and habits of grid users. This allows us to both provide better guidance to future efforts in more general solutions and to collect important data about grids' usage, like workloads, for example.

The remaining of this paper is structured in the following way. In Section A.2 we go into further details about the grid assembling problem, discussing related works and presenting our approach. We discuss how BoT applications are suitable for running with resources provided by OurGrid in Section A.3. Section A.4 describes the design of OurGrid and the network of favors model. An evaluation of the system is discussed in Section A.5. In Section A.6 we expose the future steps planned in the OurGrid development. Finally, we make our concluding remarks in Section A.7.

A.2 Assembling a Grid

In a traditional system, like a LAN or a parallel supercomputer, a user obtains access to resources by negotiating with the resources' owner the right to access them. Once access is granted, the system's administrator configures to the user a set of permissions and priorities.

Although this procedure is still used also in grid computing, due to grid's inherent wide distribution, spawning across many administrative boundaries, this approach is not suitable.

Grid computing aims to deal with large, heterogeneous and dynamic users and resources sets [Fos01]. Moreover, if we are to build large scale grids, we must be able to form them with mutually untrusted and even unknown parts. In this scenario, however, it is very difficult to an ordinary user to obtain access to more than a small set of services whose owners are known. As grid computing aims to provide access to large quantities of resources widely distributed, giving the users the possibility of accessing only small quantities of resources means neglecting the potential of grid computing.

The problem of assembling a grid also raises some issues from the perspective of the resource providers. Suppose a very simple scenario where just two institutions, A and B, want to create a grid joining their resources. Both of them are interested in having access to as many resources as possible. Also, both of them shall want some fairness in the sharing. Probably both of them will want to assure that they will not only give access to their resources to the other institution's users, but also that its users will access the other institution's resources, maybe in equal proportions. Existing solutions in grid computing allow these two institutions to define some policies in their resource sharing, creating static constraints and guarantees to the users of the grid [CFK+98; TTL03; CM99]. However, if a third institution C joins the grid, new agreements must be negotiated between the institutions and configured on each of them. We can easily see that these mechanisms are neither scalable nor flexible enough to the large scale grids scenarios.

A.2.1 Related Work

Although grid computing is a very active area of research, until recently, research efforts in the dynamic access gaining to resources did not exist. We attribute this mainly to the recentness of grid computing, that has made necessary to postpone the question of grid assembling until the technologies needed to use grids matured.

Past efforts have been made in defining mechanisms that support static access policies and constraints to allow the creation of metacomputing infrastructures across different administrative domains like in the Condor system [TTL03] and in the Computational Co-op [CM99].

Since 1984, the Condor system has used different mechanisms for allowing a Condor user to access resources across institutional boundaries. After trying to use institutional level agreements [ELvD+96], Condor was changed to a user-to-institution level [TTL03], to provide flexibility, as requested by its users. Recently, it was perceived also that interoperability with grid middlewares was also needed, and a new architecture for accessing grid resources was developed [FTF+02]. Although it has not dealt with dynamic grid assembling, the Condor project has made valuable contributions to understanding the needs of users in accessing and using the grid.

The Computational Co-op defined a mechanism for gathering sites in a grid using cooperatives as a metaphor. This mechanism allows all sites to control how much of their resources are being used by the grid and provides guarantees on how much resource from the grid the sites can use. This is done through a proportional-share ticket-based scheduler. The tickets are used by users to access both local and grid resources, obtaining priorities as they spend the tickets. However, both the need for negotiations between the owners of the sites to define the division of the grid tickets and the impossibility of tickets transfers or consumption makes the Co-op not flexible enough to environments as dynamic as grids. Moreover, just as e-cash, it depends on good cryptography infrastructure to make sure that tickets are not forged.

A recent effort related to grid assembling is the research on grid economy. Namely, the Grid Architecture for Computational Economy (GRACE) [BAG00], the Nimrod/G system [ABG02] and the Compute Power Market [BV00] are related to our work. The GRACE is an abstract architecture that supports different economic models for negotiating access to grid resources. Nimrod/G is a grid broker for the execution of parameter sweep applications that implements GRACE concepts, allowing a grid client to negotiate access to resources paying for it. The Compute Power Market aims to provide access to resources in a decentralized manner, through a peer-to-peer network, letting users pay in cash for using grid resources. An important point to note in these approaches is that for allowing negotiations between service consumers and providers using secure global currencies as proposed by Nimrod/G and Compute Power Market, an infrastructure for the secure negotiation, payments and banking must be deployed. The level of maturity of the basis technologies — as, for example, secure and well deployed electronic money — makes necessary to postpone the use of economic-

based approaches in real systems.

A.2.2 OurGrid Approach

The central point of OurGrid is the utilization of assumptions that, although more restrictive to the system's usefulness, are easier to satisfy than those of existing systems based on grid economy. Our assumptions about the environment in which the system will operate are that (i) there are at least two peers in the system willing to share their resources in order to obtain access to more resources; and (ii) the applications that will be executed using OurGrid need no quality of service (QoS) guarantees. With these assumptions, we aim to build a resource sharing network that promotes equity in the sharing. By equity we mean that participants in the network which have donated more resources are prioritized when they ask for resources.

With the assumption that there will be at least two resource providers in the system, we ensure that there will exist participants in the system that own resources which access can be exchanged. This makes possible the use of an exchange based economic model, instead of the more commonly used price based models [ABG02; BV00].

By assuming that there are no requirements for QoS guarantees, we put aside negotiations, once providers need not negotiate a product whose characteristics won't be guaranteed. Without negotiations, it becomes unnecessary that participants even agree on values for the resources allocated and consumed. This simplifies the process, once consumers don't have to verify that an agreed value was really consumed and providers don't have to assure that resources are provided as agreed.

Actually, in this way we are building the simplest form of an exchanged based economic model. As there's no negotiation, every participant does favors expecting to be reciprocated, and, in conflicting situations, prioritizes those who have done more favors to her in the past. The more a participant offers, the more it expects to be rewarded. There are no negotiations or agreements, however. Each participant accounts her favors only to herself, and cannot expect to profit from them in other way than getting other participants to make favors to her.

As there is no cost in donating idle cycles — since they will be forever lost if not consumed instantaneously —, a participant in the model can only gain from donating them. As, by our first assumption, there exists at least one other participant which is sharing her idle resources, donating implies in eventually benefiting from accessing extra resources.

As we shall see, from the local behavior of all participants, the emergent behavior of the system promotes equity in the arbitration of conflicting requests for the shared resources in the system. An important point is that the absence of QoS guarantees makes impossible to *guarantee* equity in the resource sharing. The system can't guarantee that a user will access enough resources to compensate the amount she donated to the community, because it can't guarantee that there will ever be available resources for the time needed. As such, we propose a system that aims not to guarantee, but to *promote* the resource sharing equity. Promoting equity means trying, via a best-effort strategy, to achieve equity.

The proposed assumptions about the system ease the development and deployment of OurGrid, restricting, in turn, its utility. The necessity of the participants to own resources excludes users that don't own any resources, but are willing to pay (e.g. in cash) to use the grid. Also, the absence of QoS guarantees makes impossible the advance reservation of resources and, consequently, preclude mechanisms that provide synchrony to the execution of parallel applications that need communication between tasks.

We believe, however, that even with these restrictions, OurGrid is still very useful. OurGrid delivers services that are suitable to the BoT class of applications. As stated before, these applications are relevant to many areas of research, being interesting to many users.

A.3 Bag-of-tasks Applications

Due to the independence of their tasks, BoT applications are especially suited for execution on the grid, where both failures and slow communication channels are expected to be more frequent than in conventional platforms for the execution of parallel applications. Moreover, we argue that this class of applications can be successfully executed without the need of QoS guarantees, as in the OurGrid scenario.

A BoT application can perform well with no QoS guarantees as it (i) does not need any synchronization between tasks, (ii) has no dependencies between tasks and (iii) can tolerate faults caused by resources unavailability with very simple strategies. Examples of such strategies to achieve fault tolerance in the OurGrid scenario are replication of tasks in multiple resources or the simple re-submition of tasks that failed to execute [PCB03; SCF+00]. As such, this class of applications can cope very well with resources that nei-

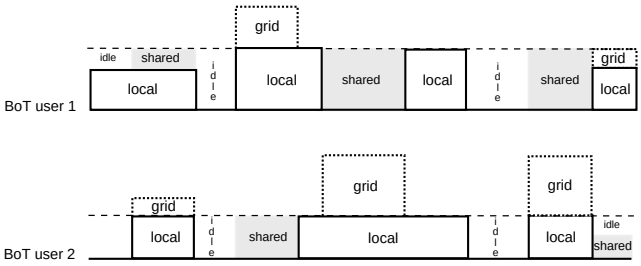


Figure A.1: Idle resource sharing between two BoT users

ther are dedicated nor can have their availability guaranteed, as failure in the execution of individual tasks does not impact on the execution of the other tasks.

Besides performing well with our assumptions, another characteristic of BoT applications that matches our approach is the work cycle of their users. Experience says that, once the application is developed, users usually carry out the following cycle: (a) plan details of the computation, (b) run the application, (c) examine the results, (d) restart cycle. Planning the details of the computation means spending the time needed to decide the parameters to run the application. Often, a significant amount of time is also needed to process and understand the results produced by a large scale computation.

As such, during the period in which a user is running her BoT application, she wants as much resources as possible, but during the other phases of her work cycle, she leaves her resources idle. These idle resources can be provided to other users to receive, in return, access to the resources of other users, when needed. An example of this dynamic for two users of BoT applications is illustrated in Figure A.1. In this figure, the users use both local resources and resources obtained from the grid — which, in this case, are only the idle resources of the other user — whenever they need to run their BoT applications. Note that whenever a user needs her own resources, it has priority over foreign users.

Another point to note is that, as resources are heterogeneous, a user might not own the resources she needs to run an application that poses constraints on the resources it needs. For example, a user can own machines running both Linux and Solaris. If she wants to run an application that can run only on Solaris, she won't be able to use all of her resources. As such, it is possible for a user to share part of her resources while consuming other part, maybe in addition to resources from other users.

In this way, we believe that expecting BoT applications users to share some of their resources in order to gain access to more resources is very plausible. As stated before, this kind of exchange can be carried out without any impact to the resources owners, because there are exchanged only resources that otherwise would be idle. In return, they get extra resources when needed to run their applications.

A.4 OurGrid

Based on the discussed approach we intend to develop OurGrid to work as a peer-to-peer network of resources owned by a community of grid users. By adding resources to the peer-to-peer network and sharing them with the community, a user gains access to all the available resources on it. All the resources are shared respecting the policies of each provider and OurGrid strives to promote equity in this sharing.

A user accesses the grid through the services provided by a peer, which maintains communication with other peers and uses the community services (e.g. application-level routing and discovery) to access them, acting as a grid broker to its users. A peer P will be accessed by *native* and *foreign* users. Native users are those who access OurGrid resources through P, while foreign users are those who have access to P's resources via other peers.

A peer is both a consumer and a provider of resources. When a peer P is making a favor in response to a request from a peer Q, P is acting as a provider of resources to Q, while Q is acting as a consumer of P's resources.

OurGrid network architecture is shown in Figure A.2. Clients are software used by the users to access the community resources. A client is at least an application scheduler, possibly with extra functionalities. Examples of such clients are MyGrid [CPC+03], APST [CHY02], Nimrod/G [AGK00] and AppLeS [BWF+96].

We plan to provide access, through OurGrid, to different types of resource. In Figure A.2, for example, type A resources could be clusters of workstations accessed via Globus GRAM, type B resources could be parallel supercomputers and type C resources could be workstations running MyGrid's UserAgent [CPC+03].

Although resources of any granularity (i.e. workstations, clusters, entire institutions, etc.) can be encapsulated in an OurGrid peer, we propose them to manage access to whole

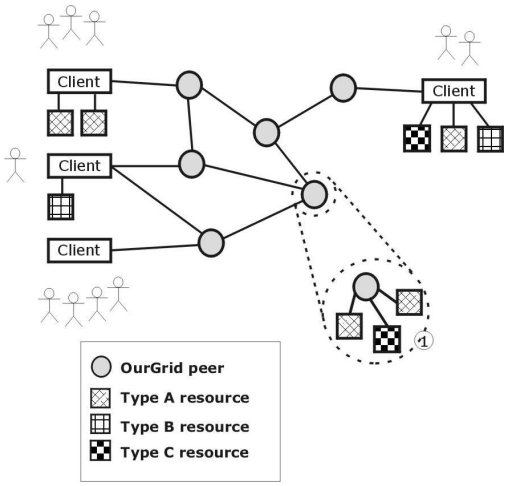


Figure A.2: OurGrid network architecture

sites instead of individual resources. As resources are often grouped in sites, using this granularity in the system will give us some advantages: (i) the number of peers in the system diminishes considerably, improving the performance of searches; (ii) the system's topology becomes closer to its network infrastructure topology, alleviating traffic problems found in other peer-to-peer systems [RF02]; and (iii) the system becomes closer to the real ownership distribution of the resources, as they are, usually grouped in sites, each with its proper set of users, owners and policies.

Finally, an OurGrid community can be part of a larger set of resources that a user has access to, and users can be native users of more than one peer, either in the same or in different communities.

In the rest of this section, we describe in details the key aspects of OurGrid design. In Subsection A.4.1 we present the model accordingly to which the resources are shared, namely, the *network-of-favors*. Subsection A.4.2 depicts the protocol used to gain access to the resources of an OurGrid community.

A.4.1 The Network of Favors

All resources in the OurGrid network are shared in a network of favors. In this network of favors, allocating a resource to a requesting consumer is a favor. As such, it is *expected* that the consumer becomes in debt with the owner of the consumed resources. The model is

based on the expectation that its participants will reciprocate favors to those consumers they are in debt with, when solicited. If a participant is not perceived to be acting in this way, it is gradually less prioritized, as its debt grows.

Every peer in the system keeps track of a local *balance* for each known peer, based on their past interactions. This balance is used to prioritize peers with more credit when arbitrating conflicting requests. For a peer p, all consumptions of p's resources by another peer p' are debited from the balance for p' in p and all resources provided by p' maintained by p are credited in the balance p maintains for p'.

With all known peers' balances, each participant can maintain a ranking of all known participants. This ranking is updated on each provided or consumed favor. The quantification of each favor's value is done locally an independently — as negotiations and agreements aren't used —, serving only to the decisions of future resource allocations of the local peer. As the peers in the system ask each other favors, they gradually discover which participants reciprocate their favors. These peers are then prioritized, based on their balances.

As a consequence, while a participant prioritizes those who cooperate with him in satisfactory ways, it marginalizes the peers who, for any reason, do not reciprocate the favors satisfactorily. The non-reciprocation can happen for many reasons, like, for example: failures on services or on the communication network; the absence of the desired service in the peer; or the utilization of the desired service by other users at the moment of the request. Free-rider [AH00] peers may even choose not to reciprocate favors. In all of this cases, the non-reciprocation of the favors gradually diminishes the probability of the peer to access the grid's resources.

Note that our mechanism of prioritizing intends to solve only conflicting situations. It is expected that, if a resource is available and idle, any user can access it. Thus, users that contribute very little or don't contribute can still access the resources of the system, but only if no other peer that has more credit requests them. The use of idle and not requested resources by peers that don't contribute (i.e., free-riders) actually maximizes the resource utilization, and does not harm the peers who have contributed with their resources.

Another interesting point is that our system, as conceived, is totally decentralized and composed of autonomous entities. Each peer depends only on its local knowledge and decisions to be a part of the system. This characteristic greatly improves the adaptability and

robustness of the system, that doesn't depend on coordinated actions or global views [BM93].

A.4.2 The OurGrid Resource Sharing Protocol

To communicate with the community, gain access to, consume and provide resources, all peers use the OurGrid resource sharing protocol. Note that the protocol concerns only the resource sharing in the peer-to-peer network. We consider that the system uses lower-level protocols to other necessary services, such as peers discovery and broadcasting of messages. An example of a platform that provides these protocols is the JXTA [JXT04] project.

The three participants in the OurGrid resource sharing protocol are clients, consumers and providers. A client is a program that manages to access the grid resources and to run the application tasks on them. OurGrid will provide part of these resources, transparently offering computational resources to the client. As such, a client may (i) access both OurGrid peers and other resources directly, such as Globus GRAM [CFK+98] or a Condor Pool [TTL03]; and (ii) access several OurGrid peers from different resource sharing communities. We consider that the client encompasses the application scheduler and any other domain-specific module needed to schedule the application efficiently.

A consumer is the part of a peer which receives requests from a user's client to find resources. The consumer is used first to request resources to providers that are able and willing to do favors to it, and, after obtaining them, to execute tasks on the resources. Providers are the part of the peers which manages the resources shared in the community and provides them to consumers.

As illustrated in Figure A.3, every peer in the community has both a consumer and a provider modules. When a consumer receives a request for resources from a local user's client, it broadcasts to the peer-to-peer network the desired resources' characteristics in a $ConsumerQuery$ message. The resources' characteristics are the minimum constraints needed to execute the tasks this $ConsumerQuery$ message is referring to. It is the responsibility of the client to discover this characteristics, probably asking this information to the user. Note that as it is broadcasted, the $ConsumerQuery$ message also reaches the provider that belongs to the same peer the consumer does.

All providers whose resources match the requested characteristics and are available (accordingly to their local policies) reply to the requester with a $ProviderWorkRequest$ mes-

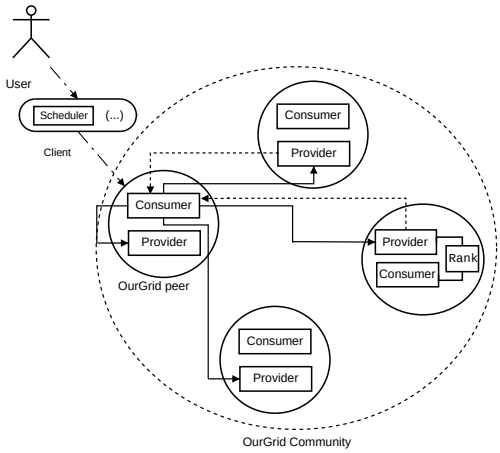


Figure A.3: Consumer and provider interaction.

sage. The set of replies received up to a given moment defines the grid that has been made available for the client request by the OurGrid community. Note that this set is dynamic, as replies can arrive later, when the resources needed to satisfy the request became available at more providers.

With the set of available resources, it is possible for the consumer peer to ask for its client to schedule tasks onto them. This is done sending a $ConsumerScheduleRequest$ message containing all known available providers. The application scheduling step is kept out of the OurGrid scope to allow the user to select among existing scheduling algorithms [CLZB00; PCB03] the one that optimizes her application accordingly to her knowledge about the characteristics of the application.

Once the client has scheduled any number of tasks to one or more of the providers who sent $ProviderWorkRequest$ messages, it sends a $ClientSchedule$ message to the consumer to which it requested the resources. As a peer represents a site, owning a set of resources, the $ClientSchedule$ message can contain either a list of ordered pairs $(task, provider)$ or a list of tuples $(task, provider, processor)$. It is up to the client deciding how to format its $ClientSchedule$ message. All tasks are sent through the consumer and not directly from the client to the provider, to allow the consumer to account its resource consumption.

To each provider P_n in the $ClientSchedule$ message, the consumer then sends a $ConsumerFavor$ message containing the tasks to be executed in P_n with all the data needed

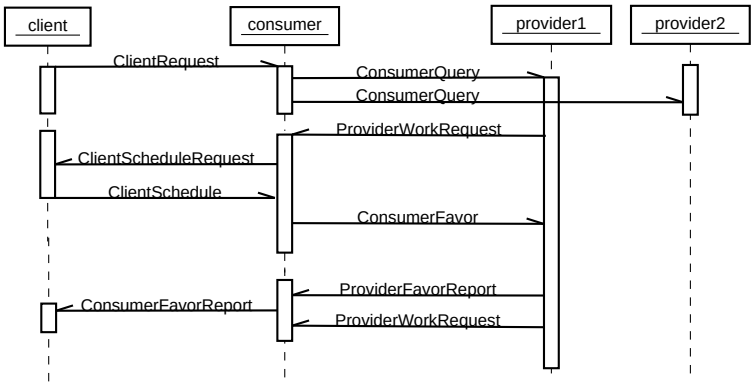


Figure A.4: Sequence diagram for a consumer and two providers interaction

to run it. If the peer who received the $ConsumerFavor$ message finishes its tasks successfully, it then sends back a $ProviderFavorReport$ message to the corresponding consumer. After concluding each task execution, the provider also updates its local rank of known peers, *subtracting* the accounting it made of the task execution cost from the balance of the consumer peer. The consumer peer, on receiving the $ProviderFavorReport$, also updates its local rank, but *adding* the accounting it made of the reported tasks execution cost to the provider balance. Note that the consumer may either trust the accounting sent by the provider or make its own autonomous accounting.

While a provider owns available resources that match the request's constraints and is willing to do favors, it keeps asking the consumer for tasks. A provider may decide not to continue making favors to a consumer in order to prioritize another requester who is upper in its ranking. The provider also decides to stop requesting tasks if it receives a message from the consumer informing that there are no tasks left to schedule or if it receives no response from a task request. Note that, after the first broadcast, the flow of requests is from the providers to the consumer. As the $ProviderWorkRequest$ messages are the signal of availability, we alleviate the consumer from the task of managing the state of its current providers.

In Figure A.4, a sequence diagram for an interaction between a consumer and two providers is shown. The provider $provider1$ makes a favor to $consumer$, but $provider2$ either is unable or has decided not to provide any resources to $consumer$.

Although an OurGrid network will be an open system, potentially comprised by different

algorithms and implementations for its peers, we present in the following sections examples of expected correct behaviors for both the provider and consumer modules of a peer. The algorithms intend to exemplify and make clearer how should a peer behave to obtain access to the community's shared resources.

Provider Algorithm

A typical provider runs three threads: the receiver, the allocator and the executor. The receiver and the allocator execute continuously, both of them access, add, remove and alter elements of the list of received requests and of known peers. The executor is instantiated by the allocator to take care of individual tasks execution and accounting.

The receiver thread keeps checking for received requests. For each of the requests received, it verifies if the request can be fulfilled with the owned resources. It does so by verifying if the provider owns resources to satisfy the request's requirements, no matter if they are available or not, accordingly to the peer's sharing policies. If the consumer request can be satisfied, the receiver adds it to a list of received requests. There are two such lists, one for requests issued by local users and another one for those issued by foreign users. This allows us to prioritize local users requests in the scheduling of tasks to the local resources. The allocator thread algorithm is shown in Algorithm 1.

While executing, the allocator thread continuously tries to satisfy the received requests with the available resources. It tries to find first a request from a local user which can be fulfilled, and if there's none, it tries the same with the community requests received.

The function *getLocalRequestRanked()*, in line labeled 2, returns the request with the specified position in the priority ranking, accordingly to a local set of policies. The policies can differ from peer to peer, but examples of local prioritizing policies would be FIFO or to prioritize the users who consumed less in their past histories. The function *getCommunityRequestRanked()*, in line labeled 5, does the same thing, but for the community requests. It must be based on the known peers balances, that serve as a ranking to prioritize these requests.

On lines labeled 3 and 6, the allocator verifies if the resources necessary specified to fulfill the request are available, according to the local policies of availability. If some request was chosen to be answered in this iteration of the main loop, the allocator decides which

```
   Data          : communityRequests, localRequests, knownPeersCredits, localPriorityPolicies
   while true do
       chosen = null ;
       /* local users' requests are prioritized over the community's */;
1      if localRequests.length > 0 then
           rank = 1 ;
           repeat
2              actual = getLocalRequestRanked( localRequests, localPriorityPolicies, rank++ );
3              if isResourceToSatisfyAvailable( actual ) then
                   chosen = actual ;
               end
           until ( chosen != null ) || ( rank > localRequests.length) ;
       end
       /* if there's no local user's request which can be satisfied */;
4      if ( chosen == null ) && ( communityRequests.lenght > 0 ) then
           rank = 1 ;
           repeat
5              actual = getCommunityRequestRanked( communityRequests, knownPeersBalances, rank++ );
6              if isResourceToSatisfyAvailable( actual ) then
                   chosen = actual ;
7                  resourcesToAllocate = getResourcesToAllocateTo( chosen );
               end
           until ( chosen != null ) || ( rank > communityRequests.length) ;
       end
       /* actually allocate resource to the chosen task */;
8      if chosen != null then
9          send( chosen.SrcPeerID, ProviderWorkRequest ) ;
           receivedMessage = receiveConsumerFavorMessage( timeout ) ;
           if receivedMessage != null then
               receivedTasks = getTasks( receivedMessage );
               foreach task in receivedTasks do
10                 execute( tasks, resourcesToAllocate );
               end
           else
11             if isRequestLocal( chosen ) then
                   localRequests.remove( chosen ) ;
               else
12                 communityRequest.remove( chosen ) ;
               end
           end
       end
   end
```

Algorithm 1: Provider's allocator thread algorithm

resources will be allocated to this request (line labeled 7) and sends a message asking for tasks to execute.

If it receives tasks to execute, it then schedule the received tasks on the resources allocated to that request. This is done on the *execute()* function, which creates a provider executor thread for each task that will be executed. The executor first sets up the environment in which the task will execute. To set up the environment means to prepare any necessary characteristics specified by the local policies of security. For example, it could mean to create a directory with restricted permissions or with restricted size in which the task will execute.

After the task is effectively executed and its results have been collected and sent back, the

```
   Data          : provider, scheduledTasks, knowPeersCredits
   send( provider, ConsumerFavor );
   unansweredTasks = scheduledTasks;
   while ( unansweredTasks.lenght > 0 ) && ( timeOutHasExpired( ) == false ) do
       results = waitProviderFavorReport( providerID );
       answeredTasks = results.getTasks( );
       removeReportedTasks( answered Tasks, unansweredTasks );
       foreach task in answeredTasks do
           if isProviderLocal ( provider ) == false then
1              usage = quantifyUsage( results );
               previousBalance = getPeerBalance( knownPeerBalance, provider );
2              updatePeerBalance( knownPeersBalance, provider, ( previousBalance + usage ) );
           end
       end
   end
```

Algorithm 2: Consumer's remote executor thread algorithm

executor must update the credit of the consumer peer. The quantifying function may differ from peer to peer. A simple example of how it could be done is to sum up all the CPU time used by the task and multiply it by the CPU speed, in MIPS. Once the peer has estimated the value of the favor it just made, it updates the $knownPeersBalances$ list, decreasing the respective consumer balance.

For simplicity, we are considering here that our provider's allocator scheduling is non-preemptive. However, it is reasonable to expect that, to avoid impacting on interactive user of the shared resources, a provider may suspend or even kill tasks from foreign users.

Consumer Algorithm

As we did with the provider, this section will discuss a simple yet functional version of a consumer algorithm. The consumer runs three threads: the requester, the listener and the remote executor. The requester responsibility is to broadcast client requests it receives as $ConsumerQuery$ messages.

After the $ConsumerQuery$ message for a given $ClientRequest$ message has been sent, the consumer listener thread starts waiting for its responses. It receives all the Provider-WorkRequest messages sent to the peer, informing that the resources are available to the Client as they arrive.

Each instance of the remote executor thread, as illustrated in Algorithm 2, is responsible for sending a set of tasks to a provider, waiting for the responses and updating the balance of this provider in the local peer. The quantification is shown on line labeled 1, and may differ from peer to peer. Examples of how it can be performed may vary from simply using

the accounting sent by the provider to more sophisticated mechanisms, such as sending a micro-benchmark to test the resource performance, collect the CPU time consumed and then calculate the favor cost as a function of both. Yet another possibility is to estimate the task size, maybe asking the user this information, and then assigning a cost based on this size to each task execution.

The provider's balance is updated on line labeled 2. Note that the usage is added to the provider's balance, while in the provider's executor it was deducted.

A.5 Evaluation

In this section we show some preliminary results from simulations and analytical evaluation of the OurGrid system. Note that, due to its decentralized and autonomous nature, characterizing the behavior of an OurGrid community is quite challenging. Therefore, at this initial moment, we base our analysis on a simplified version of OurGrid, called OurGame. OurGame was designed to capture the key features of OurGrid, namely the system-wide behavior of the network of favors and the contention for finite resources. The simplification consists of grouping resource consumption into turns. In a turn, each peer is either a provider or a consumer. If a peer is a consumer, it tries to consume all available resources. If a peer is a provider, it tries to allocate all resources it owns to the current turn consumers. In short, OurGame is a repeated game that captures the key features of OurGrid and allows us to shed some light over its system-wide behavior.

A.5.1 OurGame

Our system in this model is comprised of a community of n peers represented by a set $P = \{p_1, p_2, ..., p_n\}$. Each peer p_k owns a number r_k of resources. All resources are identical, but the amounts in each peer may be different. Each peer can be in one of two states: *provider* or *consumer*. When it is in the provider state, it is able to provide all its local resources, while in the consumer state it sends a request for resources to the community. We consider that when a peer is in the consumer state it consumes all its local resources and, as such, it cannot provide resources to the community. All requests sent by the consumers are equal, requesting as much resources as can be provided.

A peer p_k is a tuple $\{id, r, state, ranking, \rho, allocationStrategy\}$. The id field represents this peer identification, to be used by other peers to control its favor balance. As stated before, r represents p_k's amount of resources, $state$ represents the peer's actual state, assuming the *provider* or *consumer* values. The *ranking* is a list of pairs $(peer\_id, balance)$, representing the known peers ranking. In this pair, $peer\_id$ represents a known peer and $balance$ the credit or debit associated with this peer. To all unknown peers, we consider $balance = 0$. The ρ field is the probability of p_k of being a provider in a given turn of the game.

The $allocationStrategy$ element of the tuple defines the peer's resource allocation behavior. As instances of the $allocationStrategy$, we have implemented $AllForOneAllocationStrategy$ and $ProportionallyForAllAllocationStrategy$. The former allocates all of the provider's resources to the consumer that has the greatest balance value (ties are broken randomly). The $ProportionallyForAllAllocationStrategy$ allocates the peer's resources proportionally to all requesting peers with positive balance values. If there are no peers with positive balance values, it allocates to all with zero balance values and if there are no requesting peer with a non-negative balance value, it allocates proportionally to all requesting peers.

In this model the time line is divided in turns. The first action of all peers in every turn is to, accordingly to its ρ, choose its state during the turn, either consumer or provider. Next, all peers who are currently in consumer state send a request to the community. All requests arrive in all peers instantaneously, asking for as many resources as the peer owns. As our objective is studying how the system deals with conflicting requests, all consumers always ask for the maximum set of resources.

On receiving a request, each provider chooses, based on its $allocationStrategy$ which resources to allocate to which consumers, allocating always all of its resources. All allocations last for the current turn only. In the end of the turn, each peer updates its $ranking$ with perfect information about the resources it provided or consumed in this turn.

A.5.2 Scenarios

To verify the system behavior, we varied the following parameters:

- Number of peers: We have simulated communities with 10, 100 and 1000 peers;
- Peers strategies: Regarding the $allocationStrategy$, we have simulated the following scenarios: 100% of the peers using $AllForOneAllocationStrategy$, 100% using $ProportionallyForAllAllocationStrategy$ and the following combinations between the two strategies in the peers: (25%, 75%), (50%, 50%) and (75%, 25%).
- Peer probability of being a provider in a turn (ρ): We have simulated with all peers having a probability of 0.25, 0.50 and 0.75 to be in the provider state. Also, we have simulated an heterogeneous scenario in which each peer has a probability of being a provider given by a uniform distribution in the interval [0.00..0.99]. We have not considered peers with probability 1.00 of being a provider because we believe that the desire of consuming is the primary motivation for a site to join an OurGrid community, and a peer would not join it to be always a provider.
- Amount of resources owned by a peer: All peers own an amount of resources drawn from a uniform distribution in the interval [10..50]. We considered this to be the size of a typical laboratory that will be encapsulated in an OurGrid peer.

All combinations of those parameters gave us 60 simulation scenarios. We have implemented the model and this scenarios using the SimJava [HM98] simulation toolkit[2].

A.5.3 Metrics

Since participation in an OurGrid community is voluntary, we have designed OurGrid (i) to promote equity (i.e., if the demand is greater than the offer of resources, the resources obtained from the grid should be equivalent to resources donated to the grid), and (ii) to prioritize the peers that helped the community the most (in the sense that they have donated more than they have consumed). We gauge equity using *Favor Ratio* (FR) and the prioritization using *Resource Gain* (RG).

The Favor Ratio FR_k of a peer p_k after a given turn is defi ned by the ratio of the accumulated amount of resources gained from the grid (note that this excludes the local resources

[2]SimJava is available at `http://www.dcs.ed.ac.uk/home/simjava/`

consumed) by the accumulated amount of resources it has donated to the grid. More precisely, for a peer p_k which, during t turns, gained g_k resources from the grid and donated d_k to the grid, $FR_k = g_k/d_k$. As such, FR_k represents a relation between the amount of resources a peer gained and how much resources it has donated. If $FR_k = 1$, peer p_k has received from the grid an amount of resources equal to that it donated. That is, FR_k = 1 denotes equity.

The Resource Gain RG_k of a peer p_k after a given turn is obtained dividing the accumulated amount of resources used by it (both local and from the grid) by the accumulated amount of local resources it has used. As such, let l_k be all the local resources a peer p_k consumed during t turns and g_k the total amount of resources it obtained from the grid during the same t turns, $RG_k = (l_k + g_k)/l_k$. RG_k measures the "speed-up" delivered by the grid, i.e. how much grid resources helped a peer in comparison to its local resources.

Note that RG_k represents the resources obtained by a peer when it requested resources, because whenever a peer asks for grid resources, it is also consuming its local resources. Thus, we can interpret RG_k as a quantifi cation of how much that peer was prioritized by the community.

Thus, to verify the equity in the system-wide behavior, we expect to observe that, in situations of resource contention, $FR_k = 1$ for all peers p_k. We want also to verify if the peers which donated more to the community are really being prioritized. To gauge this, we will use RG_k, which we expect to be greater for the peers with the greatest differences between what they have donated and what they have consumed from the community.

Due to the considerations of this model, we can easily draw a relation between RG_k and FR_k. Consider a peer p_k, let r_k be the amount of resources it owns, t the number of turns executed, l_k its local resources consumed, d_k the amount of resources it donated to the community, i_k the resources that went idle because there were no consumers in some turns in which p_k was in the provider state and ρ_k the probability of the peer p_k being a provider in a given turn. Let us also denote R_k as the total amount of local resources that a peer had available during t turns. As such:

$$\begin{cases} R_k = t.r_k \\ R_k = l_k + d_k + i_k \\ l_k = (1 - \rho_k).R_k \end{cases} \quad \text{(A.1)}$$

From (A.1) and the definitions of FR_k and RG_k, we can derive that:

$$RG_k = 1 + \frac{\rho_k.FR_k}{(1-\rho_k)} - \frac{i_k.FR_k}{(1-\rho_k).t.r_k} \quad \text{(A.2)}$$

Another relation that is useful is obtained from the fact that the total amount of resources available in the system is the sum of all resources obtained from the grid, locally consumed and left idle for all peers. As all resources donated are consumed or left idle, no resources are lost nor created, we can state a resource conservation law as follows:

$$\sum_k R_k = \sum_k g_k + \sum_k l_k + \sum_k i_k \quad \text{(A.3)}$$

A.5.4 Results discussion

With the OurGame model, the scenarios in which we instantiated the model and the metrics to measure its characteristics presented, we shall now show the results we have obtained so far. We will divide the discussion between the scenarios in which all peers have the same providing probability ρ and those in which ρ_k is given by a uniform distribution in [0..0.99]. In each of them we will then examine how the number of peers, the providing probabilities ρ_k and the $allocationStrategy$ they were using impacted their RG_k and FR_k values, and, consequently, on the network of favors behavior and on the OurGrid resource contention.

Results for Communities in Which All Peers Have Equal Providing Probabilities

For all scenarios in which ρ_k was equal to all peers, both FR_k and RG_k converged. FR_k always converged to 1, but RG_k converged to different values, depending on the scenarios' parameters.

With all peers competing with the same appetite for resources, each peer gains back the same amount of resources it has donated to the community, that explains the convergence of FR_k. Figure A.5 shows this happening, despite variance on the amount of resources owned

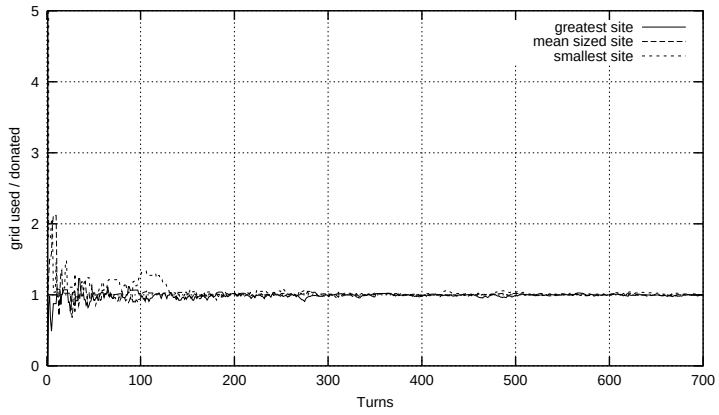


Figure A.5: FR for peers with different resource quantities in a 10-peer community using ProportionallyForAllAllocationStrategy with $\rho = 0.25$

by each peer. The three lines in the Figure are a peer with the greatest r_k, a peer with a mean value of r_k and a peer with the smaller r_k in the scenario.

Regarding RG_k, with $FR_k = 1$, from equation (A.2), we obtain that $RG_k = 1 + \frac{\rho_k}{(1-\rho_k)} - \frac{i_k}{(1-\rho_k).t.r_k}$. To facilitate our understanding, we divide the analysis of RG_k behavior in two situations: the scenarios in which there are no idle resources (i.e., $i_k = 0$) and the scenarios in which there are idle resources (i.e., $i_k > 0$).

Analytically analyzing the scenarios, we observe that in a given scenario, $i_k > 0$ happens if there is no consumer in some round. The probability of all peers $p_1, ..., p_n$ to be in the provider state is $AP = \prod_{1 \leq k \leq n} \rho_k$. Thus, the number of turns in which all resources in the community were idle in a given scenario is $IT = t.AP$. For all scenarios but the ones with 10 peers and 0.50 or 0.75 providing probabilities, $IT = 0$.

In the scenarios where $i_k = 0$, as $FR_k = 1$ we find, from equation (A.2) that $RG_k = 1 + \rho_k/(1 - \rho_k)$. As $0 \leq \rho_k < 1$, this means that $RG_k \propto \rho_k$. As such, the peers with greater ρ_k are the peers which have the greater difference between what they donated and consumed, the fact of $RG_k \propto \rho_k$ shows that the more a peer contribute to the community, the more it is prioritized. As, in this scenarios, all peers have the same ρ, however, they all have the same RG_k. For example, in a community in which all peers have $\rho = 0.50$, we found RG_k from all peers converged to $RG = 1 + 0.5/(1 - 0.5) = 2$. As can be seen in Figure A.6, for a 100-peer community.

In the scenarios in which $i_k > 0$, we also found $FQ_k = 1$ and RG_k also converged. How-

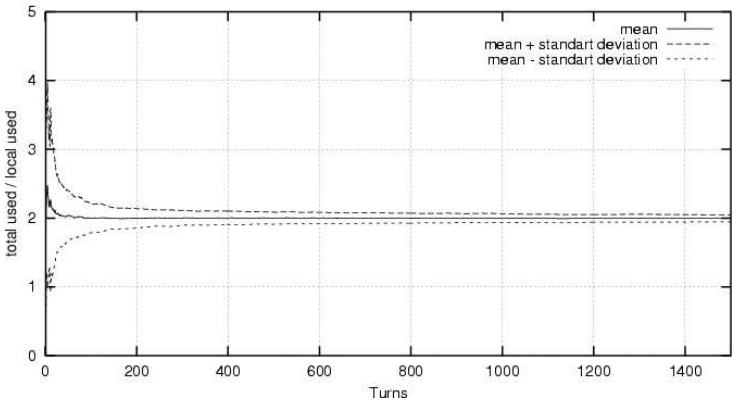


Figure A.6: RG in a 100-peer community using ProportionallyForAllAllocationStrategy and with $\rho = 0.50$

ever, we observed two differences in the metrics behavior: (i) FR_k took a greater number of turns to converge and (ii) RG_k converged to a value smaller than in the scenarios where $i_k = 0$. The former behavior difference happened because each peer took a longer time to rank the other peers, as there happened turns with no resource consumption. The latter behavior difference is explained by equation (A.3). As, for a given peer p_k, l_k is fixed, the idle resources i_k actually are resources *not consumed*. This means that g_k and, consequently, RG_k decreases as i_k increases. In short, as the total amount of resources consumed by the peers is less than the total amount of resources that were made available (i.e., both donated and idle), their RG_k is smaller than in the scenarios where all resources made available were consumed.

Finally, regarding the strategy the peers used to allocate their resources, we found that varying the strategy used by the peers did not affect significantly the metrics behavior. The number of peers in the community, on the other hand, naturally affects the number of turns needed for both metrics converge. The number of turns needed for the metrics to converge is bigger as the size of the community grows.

Results for Communities in Which Peers Have Different Providing Probabilities

After observing the effects of each of the simulation parameters in a community that had the same probability for consuming resources, we now discuss how the variation on this probability affects our defined metrics.

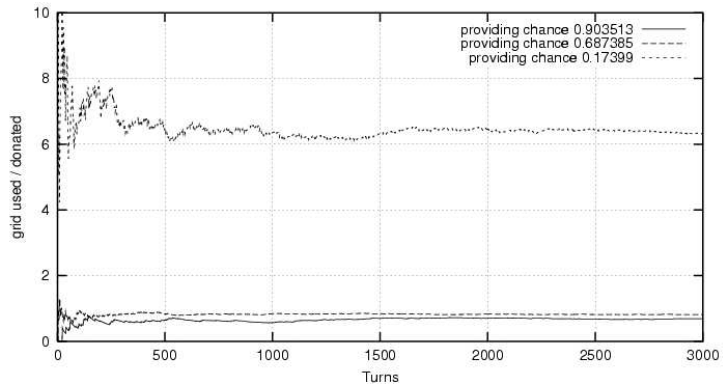


Figure A.7: FR for three peers in a 10-peer community with different providing probabilities using ProportionallyForAllAllocationStrategy

First, in the simulations of the 10-peer communities, we found that FR_k did not converge. Figure A.7 shows FR_k for three peers of a community with this number of peers and in which the providing chance ρ_k of each peer p_k is given by a uniform distribution in the interval [0.00..0.99]. As can be seen, the peer which donated less to the community — as its providing chance is smaller that of all other peers' in Figure A.7 —, obtained the greatest FR. This is easily explained if we take a look also in the values of RG_k for these peers. The RG_k behavior for the same three peers is shown in Figure A.8. Note that, for the peer with the greatest ρ, RG_k explodes the scale in its first request, after some turns providing, what gives us the almost vertical solid line in the graph. Figure A.8 shows how a peer is prioritized as it donates more resources to the community. Consequently, the peer which provided more resources is the peer with the greatest RG. Whenever it asks for resources, it manages to get access to more resources than a peer that has provided less to the community.

The peer with the lesser providing chance obtained more resources from the grid, and thus got a greater FR_k because it actually requested more resources and donated just a little bit. As the providing probabilities are static, the peers with greatest probabilities provided more and didn't asked resources often enough to make their FR_k raise. Thus, FR_k did not converge because there was not enough competition in these scenarios, and there were turns in which only peers which contributed with small amounts of resources to the community requested resources. Note that without enough competition for the resources we cannot observe the fairness of the system. Nevertheless, by observing RG_k, we still can observe

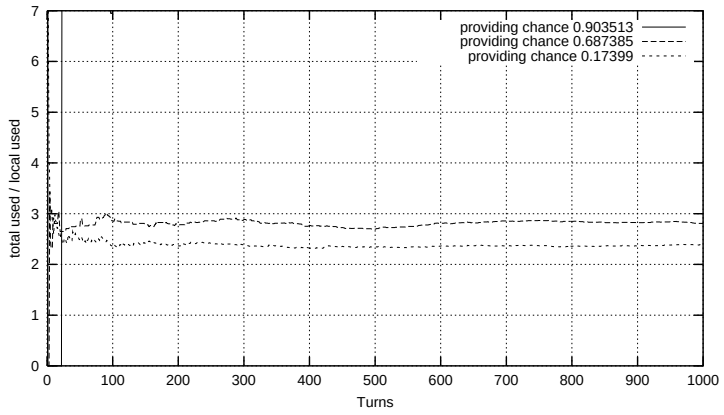


Figure A.8: RG for three peers in a 10-peer community with different providing probabilities using ProportionallyForAllAllocationStrategy

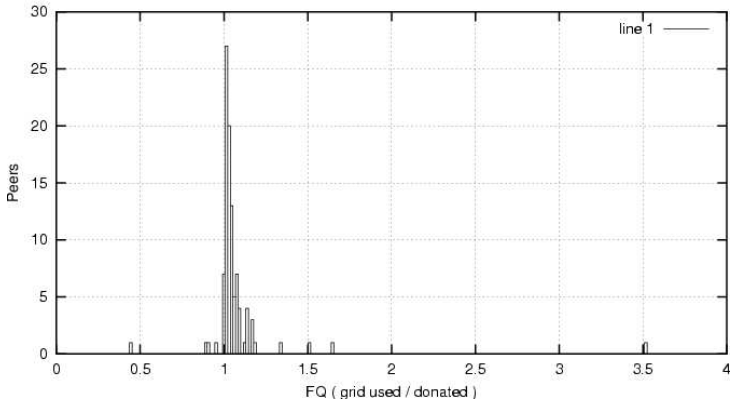


Figure A.9: FR histogram in a 100-peer community with different allocation strategies on turn 3000

how the prioritization was done, when the peers which contributed more to the community did ask for resources.

An interesting behavior we have observed is that with the growth of the community size, FR_k once again converges to 1. It happens for the 100-peer and the 1000-peers communities, in our simulations. The histogram[3] of FR_k in a 100-peer community in the turn 3000 is show in Figure A.9.

The convergence of FR_k happens due to the greatest concurrence present in greater communities. As there are more peers, there are less turns in which only peers with small ρ_i re-

[3]We opted to show a histogram due to the great number of peers in the simulation.

quest the resources of the community. As such, less peers manage to obtain FR_k as high as happened in the 10 peers scenarios. This may still happen, if there are only peers that donate very little in a sufficiently large number of turns. Nevertheless, this is not prejudicial to our objectives, as these resources could not be allocated to a peer that contributed significantly to the community.

With $FR_k = 1$ and $i_k = 0$, we again find that $RG_k \propto \rho_k$. This shows that peers which contributed more, that is, which have the highest ρ_k, were more prioritized.

We shall remark that again, our two allocation strategies did not show impact in the simulations results. As such, in the long run, peers that allocate all of their resources to the highest ranked peer perform as well as peers that allocate their resources proportionally to the balances of the requesters.

A.6 Future directions

The next steps in the OurGrid development are (i) simulating real grid users workloads on the peers; (ii) studying the impact of malicious peers in the system; and (iii) the actual implementation of OurGrid. Now that we have evaluated the key characteristics of our network of favors, simulating more realistic scenarios is needed to understand the impact of the grid environment in the model presented in this work. Peer's maliciousness is important mostly in two aspects in OurGrid: a peer consumer shall want to assure that a provider executed a task correctly and there must not be possible to exploit the community using unfair accounting.

More specifically in the malicious peers problem, to deal with the need of the consumer to assure correct task execution in unreliable providers, we plan to study both (a) replication in order to discover providers a consumer can trust and (b) the insertion of application specific verification, like the techniques described in[GM01]. To cope with the objective of making the community tolerant to peers using unfair accounting, marginalizing them, we aim to study the use of (a) autonomous accounting and (b) replication to determine if a consumer shall trust unknown providers.

We plan to start OurGrid implementation as an extension of MyGrid[4] [CPC+03; CM01] former work done at UFCG. OurGrid will be able to serve as a MyGrid resource in

[4]MyGrid is open-source and it is available at `http://dsc.ufcg.edu.br/mygrid/`

the user's grid, and will initially obtain access to resources through the already existent MyGrid's Grid Machine Interface. The Grid Machine Interface is an abstraction that provides access to different kinds of grid resources (Globus GRAM, MyGrid's UserAgent, Unix machines via ssh, etc.) and will allow OurGrid to interoperate with existing grid middleware. Interoperability is important to both take advantage of existing infrastructure and to ease the OurGrid adoption by the community of users.

A.7 Conclusions

We have presented the design of OurGrid, a system that aims to allow users of BoT applications to easily obtain access and use computational resources, dynamically forming an on-demand, large-scale, grid. Also, opting for simplicity in the services it delivers, OurGrid will be able to be deployed immediately, both satisfying a current need in the BoT users community and helping researchers in better understanding how grids are really used in production, a knowledge that will help to guide future research directions.

OurGrid is based on a *network of favors*, in which a site donates its idle resources as a favor, expecting to be prioritized when it asks for favors from the community. Our design aims to provide this prioritization in a completely autonomous manner. The autonomy is crucial to keep our system simple and not dependent on centralized services that might be hard to deploy, scale and trust.

Our preliminary results on the analysis through simulation of this design to solve the conflict for resources in a decentralized community shows us that this approach is promising. We expect to evolve the present design into a solution that, due to its simplicity, will be able to satisfy a need from real grid users *today*.

A.8 Acknowledgments

We would like to thank Hewlett Packard, CNPq and CAPES for the financial support. It was crucial to the progress of this work. We would also like to thank Elizeu Santos-Neto, Lauro Costa and Jacques Sauvé for the insightful discussions that much contributed to our work.

Appendix B

Discouraging Free-Riding in a Peer-to-Peer CPU-Sharing Grid

Nazareno Andrade[1], Francisco Brasileiro[1], Walfredo Cirne[1], Miranda Mowbray[2]

[1]*Universidade Federal de Campina Grande*

[2]*HP Laboratories Bristol*

Abstract: Grid computing has excited many with the promise of access to huge amounts of resources distributed across the globe. However, there are no largely adopted solutions for automatically assembling grids, and this limits the scale of today's grids. Some argue that this is due to the overwhelming complexity of the proposed economy-based solutions. Peer-to-peer grids have emerged as a less complex alternative. We are currently deploying OurGrid, one such peer-to-peer grid. OurGrid is a CPU-sharing grid that targets Bag-of-Tasks applications (i.e. parallel applications whose tasks are independent). In order to ease system deployment, OurGrid is based on a very lightweight autonomous reputation scheme.

Free riding is an important issue for any peer-to-peer system. The aim of this paper is to show that OurGrid's reputation scheme successfully discourages free riding, making it in each peer's own interest to collaborate with the peer-to-peer community. We show this in two steps. First, we analyze the conditions under which a reputation scheme can discourage free riding in a CPU-sharing grid. Second, we show that OurGrid's reputation scheme satisfies these conditions, even in the presence of malicious peers. Unlike other distributed mechanisms for discouraging free riding, OurGrid's reputa-

tion scheme achieves this without requiring a cryptographic infrastructure or specialized storage. [1]

B.1 Introduction

Grid computing emerged from the possibility of using large federation of resources as execution platform for parallel applications. However, nowadays, building such a federation (i.e., assembling a grid), is not a simple task. There are no largely adopted solutions that make automatic the negotiation between resource consumers and providers.

We argue that there are no largely adopted solutions because of the deployment complexity of most proposed solutions, which rely on currency-based economies for solving the general grid assembling problem [BAG00; BV00; Gri03; ABG02]. In this scenario a consumer can negotiate an arbitrarily complex provision of resources and pay for it in some currency. To deploy a solution for this scenario, an infrastructure for secure e-cash, e-banking and service auditing must be also deployed. We argue that this need is the greatest obstacle in making possible a Grid Economy nowadays.

As alternatives to the currency-based solutions, peer-to-peer grids have been proposed [ACBR03; BZH03; CFV03; TSWP03]. As resource sharing systems, they do not solve the general grid assembling problem. However, these systems can take advantage of the usual easiness of deployment of peer-to-peer systems to get in production today.

We are currently deploying a peer-to-peer grid called OurGrid [ACBR03; our03]. OurGrid solves the grid assembling problem for users of Bag-of-Tasks applications [CBS+03], those parallel applications whose tasks are independent, such as parameter sweep, massive search and Monte Carlo simulation. As in most peer-to-peer resource sharing systems, it may be possible for users to *free-ride*, consuming resources donated by others but not donating any of their own. Experience with deployed peer-to-peer systems shows that in the absence of incentives for donation, a large proportion of the peers only consume the resources of the system [AH00; RF02; SGG02]. Free-riding is a source of preoccupation because it decreases the utility of the resource-sharing system, potentially to the point of system collapse.

[1] Aceito para publicação no 13th International Symposium on High Performance Distributed Computing (HPDC-13), em Junho de 2004. / Accepted for publishing in the 13th International Symposium on High Performance Distributed Computing (HPDC-13), June of 2004.

To avoid this, OurGrid was designed to explicitly provide an incentive for peers to collaborate with the system. To provide this incentive, OurGrid uses a peer-to-peer autonomous reputation scheme, the Network of Favors. In the Network of Favors, donating resources is a favor, and each peer autonomously prioritizes the peers who have reciprocated more favors in the past. The community does not have to rely on common knowledge to be effective. That is, it is not necessary to store global reputations. Local reputations, based only on interactions directly involving the peer that stores them, can be sufficient for an effective incentive scheme. Through the autonomous behavior of its components, the system prioritizes the peers who have higher reputations, motivating sharing.

The effectiveness of the Network of Favors relies on the fact that OurGrid peers are expected to be *eager consumers*. A consumer is eager when it can benefit from whatever resources they obtain. Bag-of-Tasks applications are usually comprised of a significant number of tasks. Also, the desire to use a CPU-sharing grid, such as OurGrid, suggests a great need for computational resources. Furthermore, task replication is a simple way of improving the makespan of the application [PCB03]. This replication multiplies the demand of the application for resources. Finally, CPU is the critical resource for Bag-of-Tasks applications. Most users can just recompile their applications to the architectures usually available, if needed. Therefore, we believe that it is reasonable to assume that Bag-of-Tasks users are eager consumers, and can consume all processing power donated to them.

In the next Section, we analyze under which conditions a reputation scheme discourages free riding among eager consumers. In Section B.3, we show that OurGrid's reputation scheme satisfy these conditions, even in the presence of malicious peers. Section B.4 compare ours against related work and Section B.5 finishes up the paper with conclusions and related work.

B.2 Resource Sharing among Eager Consumers

In this section we prove a general analytical result about resource sharing among eager consumers: if the community has some mechanism (not necessarily the mechanism used in OurGrid) by which it can identify collaborators with sufficient accuracy, and known collaborators get priority in access to the resources, then it pays to be a collaborator. As a

consequence, if peers change their strategy to or from free-riding if it is in their interest to do so, then the community evolves to a state where there are no free-riders.

Let n be the number of peers in the community (we do not necessarily assume that n is large). Since we allow peers to change their strategy, the number of free-riders will vary over time. Let $f(t).n$ be the number of peers that are free-riders at time t. The other $(1 - f(t)).n$ peers are collaborators at time t, donating all their spare resources to the community.

Since the peers are eager consumers, a peer that is donated a resource always gains some positive utility as a result, no matter how large the quantity of resources it is simultaneously consuming from other sources. It is therefore reasonable to assume that the utility lost by the donor as a result of donating the resource is a fixed multiple v of the utility gained by the recipient, with $0 < v < 1$.

If $f(t) = 0$, then all peers are collaborators. If $f(t) = 1$ the community contains only free-riders, so no resources are donated and there is no incentive for any peer to remain in the community. Suppose now that $0 < f(t) < 1$ at some time t. Let $\epsilon(t)$ be the "error" probability at time t that if a collaborator has a spare resource, it will donate the resource to a free-rider. This may happen either because the collaborator cannot distinguish another collaborator to whom it can donate its resources to or because there are only free-riders requesting resources at time t. If there are no collaborators on time t, then we define $\epsilon(t) = 1$.

Now if the utility gained by the recipient of a particular resource donation is u, then the total expected utility gain to the set of collaborators as the result of the donation is $(1 - \epsilon(t)).u - v.u$, where t is the time that the donation takes place (the donor must be a collaborator, because free-riders do not donate resources), and the total expected utility gain to the set of free-riders arising from the donation is $\epsilon(t).u$. Since there are $(1 - f(t)).n$ collaborators, the expected utility gain to an average collaborator is $\frac{(1-\epsilon(t)-v).u}{(1-f(t)).n}$. Similarly, since there are $f(t).n$ free-riders, the expected utility gain to an average free-rider is $\frac{\epsilon(t).u}{f(t).n}$. Therefore, it is better to be a collaborator provided that $\frac{(1-\epsilon(t)-v).u}{(1-f(t)).n} > \frac{\epsilon(t).u}{f(t).n}$, which happens if and only if $\epsilon(t) < (1 - v).f(t)$. On the other hand, it is better to be a free-rider when $\epsilon(t) > (1 - v).f(t)$. For the case that $\epsilon(t) = (1 - v).f(t)$, collaborators and free-riders have the same expected utility.

We assume that peers will gradually change their strategies to or from free riding if it is in their interest to do so (i.e., if the expected utility with the new strategy is greater

than the expected utility with the old strategy). Therefore, if there is some t' for which $\epsilon(t) < (1 - v).f(t)$ for all $t \geq t'$, then after time t' it will always be in the interest of free riding peers to become collaborators, and so free riding will eventually die out.

B.3 OurGrid and the Network of Favors

OurGrid, a system that we are currently deploying, is a short term solution to the problem of automatic grid assembling for users of Bag-of-Tasks applications [ACBR03]. Through OurGrid, users get access to the idle processors of the community in a peer-to-peer fashion. The assembled processing power from the community forms the grid.

In OurGrid, idle processors are not explicitly advertised, but requests are propagated through the system to as many peers as possible. Messages typically have several alternative routes to reach peers, so that it is diffi cult for a malicious peer to block others' requests. Peers with idle processors can allocate the use of these processors to a requesting peer, sending the result of the calculation directly to the requesting peer.

OurGrid uses an autonomous reputation scheme called the *Network of Favors* to help peers with idle processors determine which requesting peer to donate to. A key motivation for the design of this scheme was to make it particularly lightweight and easy to implement in real systems. The central idea of the Network of Favors is that the users who are greater net contributors of processing power should get higher priority access to the spare processing power of the community. This principle acts as a guide to the apportioning of the available resources among the users currently requesting them and, thus, as an incentive for collaboration.

In the Network of Favors, allocating a processor to a peer that requests it is a favor, and the value of that favor is the value of the work done for the requesting peer. Each peer keeps a local record of the total value of favors it has given to and received from each known peer in the past. Every time it does a favor or receives one, it updates the appropriate number. The peer calculates a local reputation for each peer based on these numbers, such that a peer who has given many favors and received few will have a high reputation. The peer then uses the current reputations to decide to whom to offer a favor when it has to arbitrate between more than one requester. Thus, whenever there is resource contention, requesters with higher

reputation get priority.

Since an autonomous reputation scheme uses no information on interactions that did not directly involve the peer assessing the reputation, this reduces the options that malicious peers have to distort the reputations. Malicious strategies based on lying about the behaviour of third parties cannot be applied. One of the remaining possibilities for a malicious peer to attack the reputation system is to change identity.

In peer-to-peer networks, it is usually easy for a peer to change its identity by leaving the community and coming back as a supposed newcomer. By this method a peer with a bad reputation can easily start afresh with a newcomer's reputation. Cryptographic or other guarantees of a peer's identity generally can do little to stop this. One solution to the problem is stringent admission control. For example, a network for school children might require permission from a child's form teacher before the child could join the network, and a network for which growth was not a goal might ban all newcomers after an initial start-up phase. However, we would like our reputation scheme to impose as few barriers to growth of OurGrid as possible, so this solution is not suitable for us.

B.3.1 Calculating the Local Reputation for a Peer

In the Network of Favors, a peer A calculates $r_A(B)$, the local reputation of peer B, using just two pieces of information: the value of favors A has received from B, and the value of favors B has received from A. Let $v(A, B)$ be the total value of the processing power donated from peer A to peer B over the past history of the system. We want $r_A(B)$ to be a function of $v(A, B)$ and $v(B, A)$, and we want the value of this function to increase when B does a favor for A, to decrease when A does a favor for B, and to be zero if A has never interacted with B.

The simplest function of $v(A, B)$ and $v(B, A)$ that satisfies these conditions is:

$$r_A(B) = v(B, A) - v(A, B) \tag{B.1}$$

This is just the balance of favors that A owes to B. In a previous work [ACBR03] we have shown that through this very simple autonomous reputation mechanism, the emergent behaviour of the community is that the peers who contribute more than they consume are

prioritized, promoting equity. However, in that previous work, we did not consider the problem of malicious identity changing. However, as we shall see briefly, using equation B.1 as reputation function makes the system vulnerable by identity changing attacks.

A simple and effective solution to this problem is to require the value of $r_A(B)$ to always be greater than or equal to zero, and zero for newcomers. This gives us the slightly more sophisticated function:

$$r_A(B) = max\{0, v(B,A) - v(A,B)\} \tag{B.2}$$

This solution has been investigated by Yamagishi and Masuda's in the context of an online auction market [YM02]. In Yamagishi and Masuda's experiment, a reputation system with negative reputations was successful in the initial phase but rapidly led to a state in which there was a high number of dishonest dealers who frequently returned to the system as newcomers, whereas using a system with non-negative reputations led to steady improvements in the honesty of the dealers and the quality of goods sold. We shall see briefly that the non-negative reputation solution also makes a community robust to id-changing in OurGrid's scenario.

Using a non-negative reputation function makes it possible to avoid prioritizing malicious ID-changing peers over collaborating peers who have consumed more resources than they have contributed. However, under this new reputation function a collaborator A cannot distinguish between a malicious ID-changing peer who never donates any resources and a collaborating peer B that has donated resources to A in the past but consumed at least the same amount of resources from A. To distinguish between these types of peers, we introduce another term in the reputation function $r_A(B)$, (we call it a history term), which reflects for peer A the history of its donations from peer B. To avoid creating a difference between the reputations of long-known peers and newcomers that is too high, and therefore too costly for a newcomer to overcome, we use a sublinear function of $v(B,A)$ as the history term in $r_A(B)$: for example

$$r_A(B) = max\{0, v(B,A) - v(A,B) + log(v(B,A))\} \tag{B.3}$$

or

$$r_A(B) = max\{0, v(B,A) - v(A,B) + \sqrt{v(B,A)}\} \tag{B.4}$$

For these functions, there is a relatively large difference in the history term between peers who have not donated to A at all and peers who have donated a little, but not much difference between two peers who both have long histories of reciprocating donations from A. This corresponds to intuition on how the relative values that people attach to favors varies with the amount of past interaction with the person granting these favors. Since the history terms take large positive values for large values of $v(B, A)$, they can make it possible to identify a collaborator even if the collaborator has consumed more resources than it has donated, provided that it has donated enough in the past.

In order to calculate $r_A(B)$, we do assume that A has reliable information about $v(B, A)$ and $v(A, B)$, the value of favors received from and provided to B. Specifically, we assume that A can both (i) measure the value of a favor done by B for A; and (ii) verify that the work done was valid, i.e. that the data returned was not bogus. These assumptions are no stronger than the assumptions made for decentralized reputation schemes. To ensure the integrity of the information, A can use replication to both verify that the work was valid and that the value of the work was as reported by B. A detailed study of this approach applied to voluntary computing called *credibility-based fault tolerance* is presented by Sarmenta [Sar02]. Using this scheme, a peer replicates each task on different service providers until at least a predetermined number of returned results is equal. Also, small probe tasks can be used periodically to verify a resource donator's correctness. By acting correctly, a donator gains credibility in a consumer's view, and the consumer gains confidence about its results. In OurGrid, we intend to implement this credibility-based mechanism both to check for sabotage and to verify other peers' informed accounting. Note that implementing sabotage tolerance for returned results is necessary in any resource sharing system.

B.3.2 Evaluating the Network of Favors

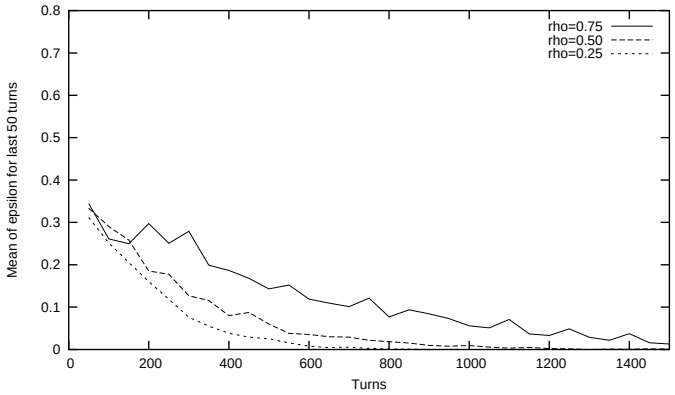
This section describes the results of some simulations that show that the autonomous reputation scheme used in OurGrid is effective at distinguishing collaborators from free-riders and promotes equitable resource sharing. We simulated the effects of all four of the reputation functions given in Subsection B.3.1. The results for the two reputation functions with history terms (Equations B.3 and B.4) were very similar, so we will not report those for the function given in Equation B.4.

We start by showing that even the very simplest reputation function – the one given by Equation B.1 — makes the amount of resources donated to non-malicious free-riders tend to zero. After this, we introduce the case when a free-rider changes identity by leaving the community and returning as a newcomer. In such scenario, the simplest reputation function cannot differentiate collaborators from free riders. We then show that the non-negative reputation schemes can successfully deal with this problem. Finally, we show that the reputation schemes with history terms have enhanced performance.

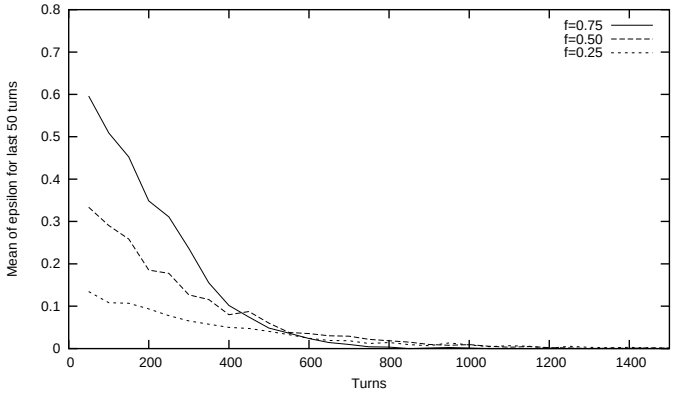
Our simulation scenario is a community of 100 peers that, in a time line divided in turns, share their resources. On each turn, each of the peers may be in consumer state with the same probability ρ. Of the hundred peers, $(1 - f).100$ are collaborators and $f.100$ are free-riders. When not in consumer state, each collaborators donates all its resources to one peer chosen among the consumers in the current turn according to their local reputation. The free-riders, on the other hand, never donate. When not consuming, they go idle.

Recall that $\epsilon(t)$ is the probability that a peer donating resources at time t will donate them to a free-rider, or 1, if there are no collaborators. This can be estimated by measuring the proportion of the available resources that were consumed by the free-riders in the simulation. Figure B.1 shows this measurement for the simulation of a 100-peer community where $f = 0.5$ and $\rho = 0.5$. As there is a wide variation in the measured values from turn to turn, we have used the value averaged over the last 50 turns in the graph. All peers are using the simple balance of the favors exchanged with other peers (as in Equation B.1) as their reputation functions. As time advances, the community identifies the free-riders, $\epsilon(t)$ gets very small, and the free-riders virtually stop getting resources.

Figures B.1(a) and B.1(b) show the behaviour of the reputation system for varying values of ρ and f. These parameters affect the time the system takes to reach the steady state where the free-riders are all identified and the early values of $\epsilon(t)$ before this state is reached, respectively. The time needed to reach the state where the community has already identified the free-riders well enough for $\epsilon(t)$ to be negligible is proportional to ρ. This happens because the community distinguishes a collaborator when it donates, and high values of ρ indicate that collaborators donate less frequently. In other words, the closer ρ is to 1, the more similar the behaviour of collaborators is to that of free-riders, and the longer it takes for the community to determine that a peer is a collaborator.



(a) $f = 0.5$ and different ρ values



(b) $\rho = 0.5$ and different f values

Figure B.1: Measurement of $\epsilon(t)$ for different values of f and ρ, where all 100 peers use $r_A(B) = v(B, A) - v(A, B)$ as reputation function

On the other hand, for larger values of f, the early values of $\epsilon(t)$ are larger, because if a collaborator donates a resource to a peer with whom the collaborator has not previously had any interactions, the probability that this peer is a free-rider is large for large f. However the value of f does not appear to signifi cantly affect the time that the system takes to identify the free-riders.

As the second step, we introduce another type of peer in the system, the *ID-changer*. This type of peer is a free-rider that assumes a new identity on every turn, making it impossible for the community to keep track of its consumption. In Figure B.2 we show how changing the 50 free-riders with stable ID in the community of Figure B.1(b) ($n = 100$, $\rho = 0.5$ and

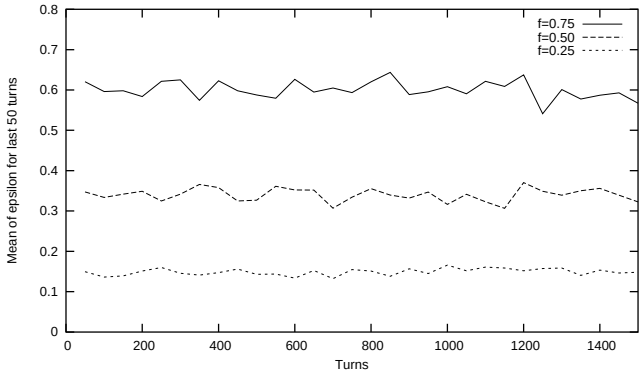


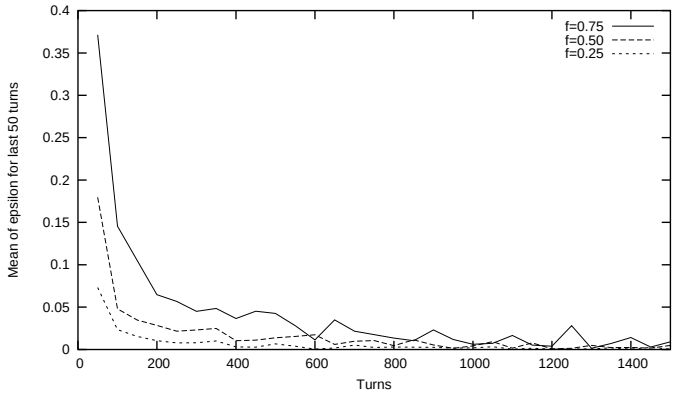
Figure B.2: Measurement of $\epsilon(t)$ in a 100-peer community with $\rho = 0.5$ and different f values, but where all free-riders are ID-changers, when all peers use the $r_A(B) = v(B, A) - v(A, B)$ as reputation function

varying f) into ID-changers alters the emergent behaviour of the system.

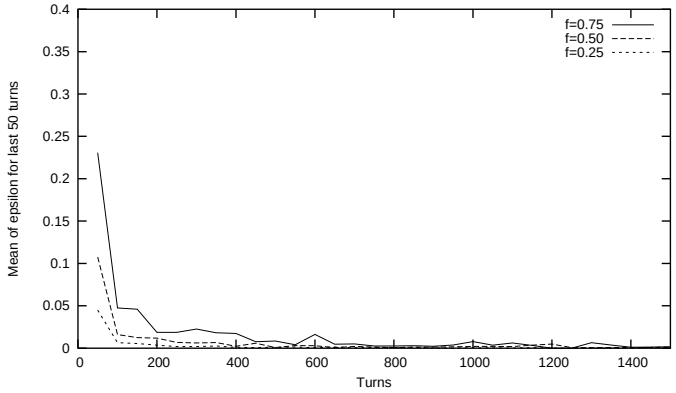
As can be seen, the capacity to distinguish the free-riders in the community greatly decreases, and $\epsilon(t)$ becomes close to f, which means that the probability that a donating peer selects a free-rider as recipient is close to the probability that a peer selected at random is a free-rider: the reputation information gives no significant help to the donating peer in distinguishing ID-changers from collaborators.

Figure B.3 shows the same scenario as Figures B.1 and B.2 ($n = 100$, $f(t) = 0.5$ and $\rho = 0.5$), but in a community where the collaborators use a non-negative reputation function with and without a history term — to be precise, the reputation functions given by Equations B.2 and B.3. Figure B.3(a) illustrates how the use of a non-negative reputation function improves the robustness of the community to the ID-changing behaviour. Adding the history term $log(v(B, A))$ further improves the ability of collaborators to identify each other, as can be seen in Figure B.3(b).

Another interesting effect of using non-negative reputations is that ρ does not significantly affect the behaviour of ϵ in communities that use this kind of reputation. We believe that this happens because, in contrast to systems that use positive and negative reputations, free-riders (and ID-changers) cannot have a reputation that is higher than that of a collaborator, and thus collaborators are more easily differentiated. Moreover, all it takes for a provider to not donate to free-riders in a turn it that one collaborator among the consumers



(a) $r_A(B) = max\{0, v(B,A) - v(A,B)\}$

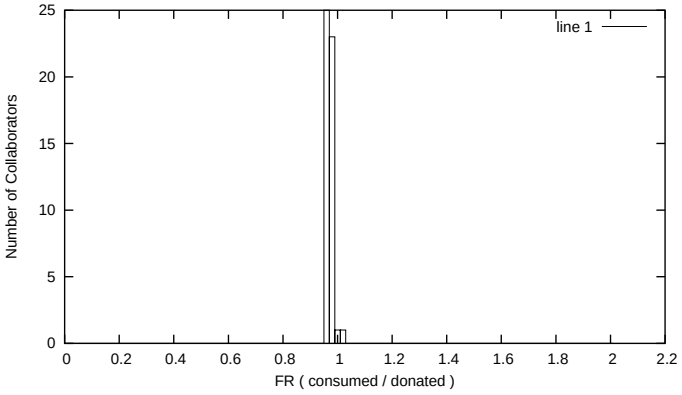


(b) $r_A(B) = max\{0, v(B,A) - v(A,B) + log(v(B,A))\}$

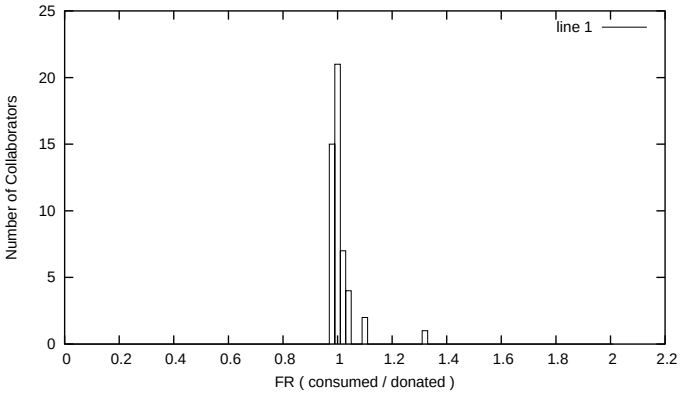
Figure B.3: Measurement of $\epsilon(t)$ for two 100-peer communities with $\rho = 0.5$ and different f values. In the first community, peers use a simple non-negative reputation function. In the second they use a non-negative reputation function with history term $log(v(B, A)$, achieving a better performance.

is distinguished. This condition seems to be easily satisfied for any value of ρ.

Finally, we have investigated the equity of the Network of Favors using the discussed reputation functions. To do this, we have introduced different-sized peers and used a metric called *Favor Ratio* (FR). In the following results, each peer owns an amount of resources given by the uniform distribution $U(1, 19)$. When donating, the peer still donates all of its resources. The Favor Ratio for a given peer is defined as the ratio of the total amount of resources it has consumed from the community to the total amount of resources it has



(a) $r_A(B) = v(B, A) - v(A, B)$



(b) $r_A(B) = max\{0, v(B, A) - v(A, B)\}$

Figure B.4: FR measured for all collaborators in two 100-peer communities where $f = 0.5$ and $\rho = 0.5$, after 3000 turns. In (a), peers use the positive and negative reputation function. In (b), they use a simple non-negative reputation function. In both FR converges to approximately 1, denoting equity.

donated to the community. Thus, if we have $FR = 1$, there is equity of resource distribution. Figure B.4(a) shows that for a 100-peer community with no ID-changers, using the simple reputation function given by Equation B.1 promotes equity. In turn 3000, FR converged to approximately 1.

In Figure B.4(b) we show that the Network of Favors still promotes equity when peers use the non-negative and non-negative-with-history reputation functions given by Equations B.2. The histograms show the distribution of FR for all collaborators in the community at

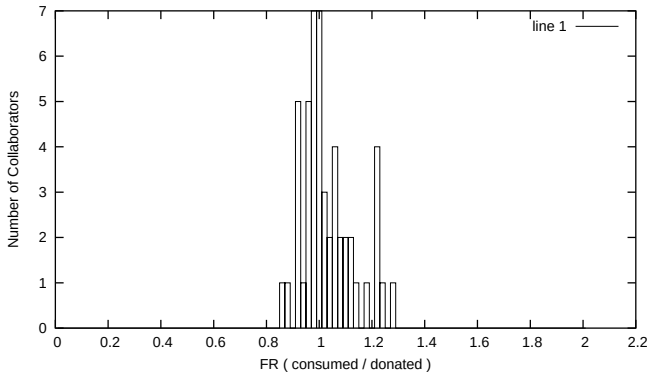


Figure B.5: FR measured for all collaborators in two 100-peer communities where $f = 0.5$ and $\rho = 0.5$, after 3000 turns. All peers use a non-negative reputation function with $log(v(B, A)$ as a history term. FR has a loosed convergence due to the history term.

the end of 3000 turns of simulation. We observe that FR in both cases converges to one, so equity still holds.

Figure B.5 shows the histogram for FR in a community where peers use the reputation function with a history term. FR has a looser convergence than in Figure B.4(b), where the community uses the reputation function without a history term. Also, the histogram of FR does not change signifi cantly for communities with reputations that consider history even if we run the simulation for larger number of turns. This happens because when the history term is introduced, the peers prioritize collaborators with whom they have a long history of interactions over consumers with greater balances with whom they have had less interaction. That is, peers start creating long-term relationships of trust and preference, trading some of the equity of the system for more stable relationships. Nevertheless, in the scenarios investigated, this trade-off has not signifi cantly affected the equity of the network of favors.

In our simulations we have assumed that peers do not change their strategies from collaborator to free rider, or from free rider to collaborator. If they did change their strategies, the value of f would not be fi xed, but would vary according to the number of free riders. Nevertheless, in all the scenarios we simulated with the three non-negative reputation functions, the measured value of $\epsilon(t)$ remained under $f/3$ after turn 100; so that for fi xed f and $v < 2/3$, we have $\epsilon(t) < (1 - v).f$ for all $t > 100$. The analysis in Section B.2 implies that in that case, free riders have lower expected utility than collaborators after turn 100. Since

OurGrid is very lightweight and a peer can preempt a guest task at any moment, we expect v to be close to zero (and certainly smaller than 2/3). After higher numbers of turns there is an advantage to collaborators for values of v even greater than 2/3.

Now consider what happens using the same reputation functions if peers do change strategies to maximize to their expected utility. It might take slightly longer for the system to identify a peer as a collaborator if the peer had originally been a free rider. However, allowing for this, at some time after the first 100 turns in such a system we expect $\epsilon(t)$ to be less than $(1 - v).f(t)$ and to remain less than $(1 - v).f(t)$ for all subsequent turns, since this holds after 100 turns for all values of f in our simulations with fixed f. It follows from Section B.2 that free riding dies out.

This argument supports our claim that if peers in OurGrid change their strategies according to their own economic interest, then free-riding will die out. Moreover, as there is equity in the division of resources, it is in the best interest of all collaborators to donate the largest amount of resources they can. As they get approximately the same amount of resources they donate, and the cost of donation is smaller than the utility gained by resource consumption, the more resources they donate, greatest is the utility they obtain from the system.

B.4 Related Work

B.4.1 Grid Resource Sharing

One existing approach to promoting equitable distribution of resources over a network is to use market-based mechanisms. In a Grid Economy [BAG00; BV00; Gri03; ABG02], resources are bought and sold over a Grid network. The currency may be convertible into real money, or may be an artificial currency only usable within the Grid. Market-based resource allocation requires secure systems for auditing and payment. Although there is currently a large amount of effort aimed at producing such systems for Grid interactions, there is not yet a stable solution that has been tested for security over a long period – and when there is one, it will require more infrastructure (for a start, a cryptographic infrastructure and some quality of service guarantees for communication) than we are assuming in this paper.

As an alternative for the initial complexity of the economy-based proposals, efforts

are being putted on the development of peer-to-peer grids [ACBR03; BZH03; CFV03; TSWP03]. Although they don't aim to solve the general problem of grid assembling, these systems can take advantage of the usual easiness of deployment of peer-to-peer resource sharing systems to have solutions in production today. The Condor [BZH03] and Triana [TSWP03] projects have proposed peer-to-peer grids, but have not considered the problem of providing incentives for resource donation, relying solely on the altruism of the system's participants. Chun et al. propose an architecture for secure resource peering based on tickets exchange [CFV03]. This architecture, however, assumes a cryptographic infrastructure and the establishment of relations of trust between peers to allow resource access. We propose, with OurGrid [ACBR03], to have a solution that is lighter and simpler to deploy in the context of Bag-of-Tasks applications. We expect to, in the subproblem of grid assembling considered, provide the simplest possible solution to promoting incentives for resource sharing. We believe this simplicity is key to make possible a wide adoption of our solution.

B.4.2 Peer-to-Peer Reputation Schemes

A reputation scheme for a peer-to-peer system is a way of recording information about past behaviour of peers, for use as a guide to other peers. The information may be derived from objective facts, or the subjective impressions recorded by other peers, or a combination of these.

For decentralized peer-to-peer systems, we tend to have distributed reputation schemes. In this type of scheme, the reputation information is distributed through different parts of the system. For example, in P2PRep [DdVPS03] each peer stores information about their own interactions with other peers, and in EigenRep [KSGM03] each peer stores local reputation values and in addition random peers store global values derived from multiple local values. In a distributed reputation scheme, a peer can retrieve all the information from the system concerning a given peer, using a retrieval protocol.

A challenging issue that a retrieval protocol must deal with is guaranteeing that the information gathered about peers is reliable, as malicious peers may tamper with the information they store. To assure the reliability of this information, P2PRep relies on voting for gathering opinions about a peer, heuristics to find clusters of potential malicious voters, and on an underlying cryptographic structure to verify the identities of the peers involved in a transac-

tion. Alternatively, in EigenRep some replicated mother peers compute and store a global reputation value for a peer. The mother peers find the peers they must keep track of, and are found by peers who need information, through a distributed hash table.

In contrast, we argue that for settings where there is eager consumption, it is possible to circumvent the need to provide such guarantees by not aggregating a global reputation value for a peer. Instead, collaboration can be efficiently promoted using an *autonomous* reputation scheme. In such a scheme, peers can only access reputation information involving peer-to-peer interactions in which they themselves have participated. This information is stored locally by the peer, so is quick to retrieve. The reputation of a given peer will in general be different in the eyes of different peers, based on their own past interactions with the peer, and there is no attempt to reconcile, average, or combine these local reputations to create a global assessment. Because the system only uses local reputations, there is no potential scalability issue arising from the retrieval and/or storage of global reputations. Moreover, as there is no need for mechanisms to ensure the integrity of information received from other peers about their interactions with third parties, such as a cryptographic infrastructure or a specialized storage infrastructure, the scheme is also lightweight.

B.5 Conclusions and Future Work

We have shown that an autonomous reputation scheme can be sufficient to promote equitable sharing of resources in OurGrid, a peer-to-peer community of eager consumers. In particular, it discourages free-riding, and can successfully deal with free-riders who change identity to try to fool the system. Our scheme is very lightweight and does not require centralized storage or a deployed cryptographic infrastructure. The only implementation issue that we have identified as potentially imposing difficulties for autonomous reputation schemes is sabotage tolerance, which is in fact an issue for any resource sharing system.

Eager consumption is a key property of our scenario that allows autonomous reputation schemes to be effective. An eager consumer tries to consume resources from as many other peers as possible. As each consumer will potentially interact with all other peers in the system, eager consumers can gain first-hand information about the behavior of other peers more quickly and easily than would be the case for a system where a peer contacts only a

few service provider at a time, as is the case for some non-eager resource sharing.

However, we suspect that autonomous reputation schemes also work under resource contention (a condition weaker than eager consumption). If peers are not eager consumers (i.e. there is a limit on the amount of resources that a consuming peer could profitably use) then peers might be able to gain all the resources they could use without building up their reputations through donation. Whether this happens or not seems to depend on the contention of the popular service providers. If the service providers can cater for only a few clients, this may incentive the clients to build up a reputation in order to "stand up from the crowd". We intend to investigate further this relation between among reputation and resource contention.

We have also analyzed how to calculate the local reputation in an autonomous reputation scheme. We have shown how using non-negative reputation functions makes the system robust to malicious identity changing, and that adding a sublinear history term can improve further the system's ability to marginalize free-riders. However, further research is needed on role of the reputation function in the formation of long-term trust relationships between peers. Future work also includes finishing the deployment of OurGrid in a 7-site grid called Pauá, which is expected to be complete by April 2004.

Appendix C

When Can an Autonomous Reputation Scheme Discourage Free-riding in a Peer-to-Peer System?

Nazareno Andrade[1], Miranda Mowbray[2], Walfredo Cirne[1], Francisco Brasileiro[1]

[1]*Universidade Federal de Campina Grande*

[2]*HP Laboratories Bristol*

Abstract: We investigate the circumstances under which it is possible to discourage free-riding in a peer-to-peer system for resource-sharing by prioritizing resource allocation to peers with higher reputation. We use a model to predict conditions necessary for any reputation scheme to succeed in discouraging free-riding by this method. We show with simulations that for representative cases, a very simple autonomous reputation scheme works nearly as well at discouraging free-riding as an ideal reputation scheme. Finally, we investigate the expected dynamic behavior of the system. [1]

[1]Aceito para publicação nos anais do 4th International Workshop on Global and Peer-to-Peer Computing, Abril de 2004. / To be published in the 4th International Workshop on Global and Peer-to-Peer Computing, in April of 2004.

C.1 Introduction

Peer-to-peer systems can be an effective and robust way of sharing resources. However, the effectiveness of several existing peer-to-peer systems is diminished by widespread free-riding [AH00; RF02; SGG02]. A peer that is a free-rider consumes resources donated by others but does not donate any resources itself. If there is a nonzero cost of donation, and the system does not discriminate between free-riders and other peers, then peers have an economic incentive to become free-riders, thus reducing the resources available for donation in the community, and diminishing the utility of the system as a whole.

One potential solution is to introduce a reputation scheme to the system. The interactions between peers affect their reputation in a way designed so that free-riders are unlikely or unable to build up a high reputation. When a peer has a resource to donate, and there are several peers requesting this resource, the peers with higher reputation are given priority. The idea is that the advantage that this gives to peers who donate resources may be enough to overcome the disadvantage given by the cost of donation.

This use of reputation differs from the classic use of reputation [RZFK00] to enhance the quality of transactions in peer-to-peer systems such as eBay [eBa03], or to marginalize untrustworthy peers as in the systems surveyed by Ooi et al. [OLK03]. If reputation is used to discourage free-riding then the choice to interact with a peer with high reputation is made in order to reward the peer for its previous behaviour, rather than to enhance the expected quality of the immediate transaction.

We are particularly interested in the circumstances under which it is possible to discourage free-riders using an *autonomous* reputation scheme. In an autonomous reputation scheme, peers use only local information to prioritize other peers. As such, they can only access reputation information involving peer-to-peer interactions in which they themselves have participated. The reputation of a given peer will in general be different in the eyes of different peers, and there is no attempt to reconcile these local reputations to create a global assessment. As a result, autonomous reputation schemes are relatively simple to implement, and do not require a cryptographic infrastructure or centralized storage to guarantee integrity of data retrieved from other peers, as is the case for some other reputation schemes that assign a single global reputation value to a peer. Autonomous reputa-

tion schemes are used in several peer-to-peer resource-sharing networks [ACBR03; Coh03; eMu03; Gro03].

An alternative way of using a reputation scheme to discourage free-riders would be not to give peers with low reputation low priority access to donated resources, but to refuse to donate resources altogether to peers with reputation below some chosen limit: if a peer had resources to donate but only peers with low reputation requested them, then the resources would remain undonated. However, this alternative would cause bootstrapping problems for an autonomous reputation scheme, because a new collaborator entering a system with autonomous reputation can only show that it is not a free-rider by donating resources, and can only detect that another peer is not a free-rider by being donated resources by that peer. Hence in this paper we consider the effect when the reputation scheme is used to prioritize donations rather than to ban certain kinds of donation completely.

In this paper we explore the design space for a peer-to-peer system in which it is possible to discourage free-riding by prioritizing resource donations using an autonomous reputation scheme.

In [ACBR03] we described an extremely lightweight autonomous reputation scheme. This was designed to promote equitable resource sharing in OurGrid, a peer-to-peer system that we are currently developing for sharing CPU cycles for bag-of-tasks applications [CBS+03]. For this autonomous reputation scheme, the reputation of peer $P1$ in the eyes of peer $P2$ is equal to the total value of resources that $P2$ has donated $P1$, minus the total value of resources that $P1$ has donated $P2$ – or is zero if this value is negative. Through simulations of representative cases, we will demonstrate the effectiveness of this reputation scheme in discouraging free-riders when the system does not exhibit eager consumption, that is, when peers in consuming state have a limit on the amount of resources that they can use with positive utility. Non-eagerness is realistic if the resource being shared is for example CPU time for applications that are not easily parallelizable, or is access to a particular software application.

We define a free-rider as a peer that does not contribute resources to the system, and a *collaborator* as a peer that does contribute resources. We say that *the system works* at time t if at that time there is a disincentive for collaborators to change their strategy to free-riding: in other words, if the expected utility for a collaborator is greater than the expected utility the

collaborator would have if it changed strategy. Free-riders always have an expected utility at least as great as the expected utility that they would have outside the system, and so if the system works then the expected utility for collaborators in the system is greater than their expected utility if they left the system.

In this paper we assume that peers in consuming state can be donated resources by any collaborator in donating state. This implies that the resources are interchangeable, which can be the case if the system shares generic CPU time, or bandwidth, or storage. However the analysis of this paper may not extend to peer-to-peer systems sharing less generic resources, such as data files, because a peer requesting a specific file will not in general be able to receive it from any peer currently donating resources, only from those peers currently donating resources that have a copy of the file. However, measurements of large scale peer-to-peer file-sharing systems[LRW03; SGG02] have found that a large percentage of all requests are for a relatively small number of files. For each one of these popular files, we can consider the peer-to-peer virtual subsystems consisting of requests for and donations of the file. Within each of these subsystems the resources are interchangeable. It is intuitively reasonable that if each of these virtual subsystems satisfies conditions that allow a particular reputation system to drive out free-riding, then by using the reputation system for the prioritization of donations in the file-sharing system as a whole it should be possible to discourage peers with typical resource requirements from free-riding. A more precise analysis of the circumstances in which a reputation scheme can be used to discourage free-riding in a file-sharing system is beyond the scope of this paper.

The rest of this paper is structured as follows. After a brief discussion of related work, we describe our autonomous reputation scheme and the design parameters for our system. Next we analyze the conditions on these parameters for the system to work at a fixed time, and use this analysis to predict system behaviour for some representative scenarios. We then simulate these scenarios for our reputation scheme to check that the predictions are met, and compare the performance of our scheme with that of an ideal reputation scheme in the simulated scenarios. Finally, we investigate the dynamic behaviour of the system if peers change their strategies according to their own economic interest.

C.2 Related Work

There recently has been an increasing amount of research in the area of reputation schemes for peer-to-peer networks, particularly for file-sharing networks. However, most of this research is on schemes that are not autonomous. For analyses of the effect on free-riding of various non-autonomous reputation schemes, see for example [Del03; KW03; RRSF03].

The file-distribution system BitTorrent [Coh03] uses an autonomous tit-for-tat mechanism to decide to whom to upload, at what bandwidth. However BitTorrent does not use long-term reputation records, because the community it serves is very dynamic and long-term peer relationships are unlikely.

Autonomous reputation schemes are used in the peer-to-peer file-sharing networks eMule [eMu03] and GNUnet [Gro03], but we are not aware of any analytical studies of the effect of these schemes on the amount of free-riding in these networks.

One previous paper, by Lai et al., [LFSC03], analyses the effect on free-riding of an autonomous reputation scheme. In the scheme analysed, if peer A receives a request from peer B, peer A has made $r > 0$ requests to B in the past, and B has cooperated with c of these, then A cooperates with B's current request with probability c/r. If resource contention is possible, Lai et al's scheme may result in some available resources being wasted. For example, suppose that an available resource is requested by just one peer, but some past requests from the resource owner were rejected by that peer because they were for contended resources. Then the available resource may not be donated.

C.3 System description

In this section we describe our autonomous reputation scheme, and the design parameters for the peer-to-peer resource-sharing system that uses it.

C.3.1 The autonomous reputation scheme

In our autonomous reputation scheme, which was introduced in [ACBR03], each peer $P1$ keeps a local record of $V(P1, P2)$ and $V(P2, P1)$ for each peer $P2$ with which it has

interacted, where $V(P1, P2)$ is the total value of resources that have been donated from $P1$ to $P2$ in the past. Each time it makes or receives a donation from $P2$, $P1$ updates this record, and recalculates its local reputation score for $P2$, which which is equal to $max\{V(P2, P1) - V(P1, P2), 0\}$. We write $r_{P1}(P2)$ to denote this score.

When peer $P1$ has spare resources that are requested by more than one other peer, it uses its local reputation scores to prioritize its donations of these resources, giving highest priority to satisfying the requests of the requesters P for which $r_{P1}(P)$ is largest.

As an illustration, suppose that peers $P1$, $P2$, $P2$ have not interacted before. We have $r_{P1}(P2) = r_{P1}(P3) = 0$. If $P2$ donates a resource with value R to $P1$, $r_{P1}(P2)$ will increase to R (whereas $r_{P1}(P3)$ will remain zero). If then $P1$ has a spare resource and has to choose between a request for that resource from $P2$ and a request from $P3$, $P1$ will choose to donate the resource to $P2$. $P1$ would make the same decision if $P3$ had interacted with $P1$ before but $r_{P1}(P3)$ was smaller than R.

C.3.2 System model

We consider a peer-to-peer system comprised of a set of collaborators and free-riders. At a fixed time t, a peer can be either in consuming or in non-consuming state. When in non-consuming state, collaborators donate their resources, while free-riders go idle. The design parameters that we consider for the peer-to-peer system are:

- Eagerness. We assume that for each peer there is a maximum value $C > 0$ of the utility of resources that can be consumed during a unit time interval when the peer is in consuming state. Thus, C limits the amount of resources that can be useful for a peer. The value of C is fixed for a given peer, but may vary between peers, with average value $\overline{C}$.

- The probability ρ of a peer being in consuming state. We assume that at a given time each peer has an independent probability ρ of being in consuming state.

- Cost of donation. The utility lost to the donator as a result of donation is a constant v times the utility gained by the recipient as a result of the donation, with $0 < v < 1$. If resources are available for donation but are not donated, no utility cost associated with these resources is incurred by the resource owner.

- Value of donation. When a collaborator is not in consuming state, it has resources of value D available to donate to the system. Note that D models performance as well as theoretical capacity. For example, if a peer has 1000 spare CPU cycles to donate, but its performance is bad and in practice the value it delivers when donating 1000 cycles is only as good as if it had no performance problems and donated 800 cycles, then D for the peer will be 800, not 1000. We assume that the value of D is fixed for a given peer, but can vary between different peers, with average value $\overline{D}$.

- The proportion f_t of peers that are free-riders at time t. The value f_t lies between 0 and 1. At time t, $N.f_t$ peers will be free-riders and $N.(1 - f_t)$ collaborators, where N is the total number of peers in the system.

For our analysis and simulations we will assume that all the values of the variables other than f_t are fixed over time.

The protocol for donation of resources is that collaborators that are not in consuming state donate all the resources that they have available as long as there are peers in consuming state prepared to consume them. Peers with high reputation are given priority in donations. We assume that the granularity of resources is low enough that a donating peer with at least as many spare resources as a consuming peer requests is able to give exactly the amount of resources requested, if the donating peer wishes to do so. Any resources left over, after all peers in consuming state have been donated the maximum amount of resources that they are prepared to accept, are not donated. Collaborators with resources to donate at a particular time do not have to donate them all to the same peer: they can donate resources to several different requesting peers. Free-riders that are not in consuming state stay idle.

C.4 Analysis

In this section we calculate the values of design parameters for which an ideal reputation system succeeds in discouraging free-riding, and use approximations to give predictions for the behaviour of sample scenarios.

Recall that we say that the system works at time t if at that time there is a disincentive to collaborators to change their strategy to free-riding. Define the *advantage to collaborators*

at time t as the expected utility gain to a collaborator as a result of being in the system minus the expected utility gain to a free-rider. This is a measure of how much free-riding is discouraged at time t. It will in general be a function of f_t. The system works at time t with $f_t = f$ if and only if either $f \in (0, 1)$ and the advantage to collaborators is positive at $f_t = f$, or $f = 0$ and the limit of the advantage to collaborators as $f_t \to 0$ is positive.

Initially we pick a fixed time t, and calculate whether the system works at that time. Later on (in Section C.6) we will discuss the dynamic behaviour of the system.

C.4.1 Analysis for fixed time

For this subsection we will assume that the system uses an ideal reputation scheme that is able to identify free-riders perfectly, that is, any free-rider always has a lower reputation than any collaborator.

Suppose at a fixed time t the total value of resources offered for donation is x_d (and that this is greater than zero), the total value of resources requested by collaborators in consuming state is x_c, and the total value of resources requested by free-riders in consuming state is x_f. We distinguish three cases, a *famine* of donations, a *glut* of donations, and the *middle* case.

The condition for famine is $x_d \leq x_c$. If this holds, then free-riders receive no donations, so gain utility zero by being in the system, whereas the set of collaborators gains a total utility $(1 - v).x_d > 0$ by being in the system. Therefore the advantage to collaborators is positive, and the system works at time t.

The condition for glut is $x_d \geq x_c + x_f$. If this holds, then all peers who make a request at time t will be donated all the resources they request. The expected utility gain for a peer resulting from the resources it is donated depends on C, but does not depend on whether the peer is a collaborator or a free-rider. On the other hand, a collaborator has an expected utility cost resulting from the resources it donates. So a collaborator can increase its overall expected utility by changing its strategy to free-riding. Therefore the advantage to collaborators is negative, and the system does not work at time t.

The condition for the middle case is that there is neither famine nor glut, ie. $x_c < x_d < x_c + x_f$. In this case the total utility gain by the set of collaborators is $x_c - v.x_d$, and the total utility gain by the set of free-riders is $x_d - x_c$. If $x_c \leq v.x_d$, then clearly the advantage to collaborators is non-positive and the system does not work at time t. Suppose $x_c - v.x_d$ is

positive and $f_t \in (0, 1)$. Then the advantage to collaborators is

$$\frac{(x_c - v.x_d)}{(1 - f_t).N} - \frac{(x_d - x_c)}{f_t.N} \tag{C.1}$$

which is a monotonically increasing function of f_t that tends to minus infinity as $f_t \to 0$. So the system does not work at time t for $f_t = 0$, and works for $f_t \in (0, 1)$ if and only if the value of this function is positive.

So far we have assumed that there is non-eager consumption. But a similar argument can be used if there is eager consumption. Eager consumption can be modeled by putting $x_c = \infty$. This implies that the condition for famine holds, whatever the values of the other variables, and hence the system works if there is eager consumption.

In this subsection we have not assumed that the allocation of resources to requesting collaborators favours those collaborators who have donated more. However, supposing it does, if there is famine and a collaborator acquires some new spare resources additional to its original resources of value D, then the collaborator has an incentive to donate these new resources to the community, provided that doing so will not move the system out of the famine condition.

We now use the results of this analysis to pick some representative scenarios for the system parameters, and make predictions for the behaviour of the system for these scenarios.

C.4.2 Predictions for sample scenarios

The mean values of x_d, x_c and x_f can be expressed in terms of the design parameters as $(1 - \rho).\overline{D}.(1 - f_t).N$, $\rho.\overline{C}.(1 - f_t).N$, and $\rho.\overline{C}.f_t.N$ respectively. We can estimate whether the system will work or not at a fixed time for a given set of parameter values, by determining whether the system will work for the mean values of x_d, x_c and x_f. This is only an estimate, because the actual values fluctuate statistically about these values, but this is a reasonable approximation to make because small changes in these values will result in small changes in the utilities we calculate. (The approximation is less accurate if D varies widely between peers.)

The scenarios we choose are the ones where the parameter values satisfy $\overline{D} = 10$, $C = 9D$ for each peer, $C = D$ for each peer, or $C = D/10$ for each peer (recall that D may

vary from peer to peer); $\rho \in \{0.1, 0.5, 0.9\}$; $f_t \in \{0.25, 0.5, 0.75\}$; and $v \in \{0.1, 0.4\}$. This makes a total of 3 x 3 x 3 x 2 = 54 sets of parameter values.

We have chosen these values to be realistic, to include both low and high realistic values, and to include some scenarios where the mean values of x_d, x_c and x_f are on the borderline between different cases.

Our prediction, using the estimate given by taking the mean values and applying the analysis of the previous subsection, is that among these 54 sets of parameter values, assuming perfect identification of free-riders, the system will work just for the 36 sets of parameter values that satisfy $C = 9D$, or $C = D$ and $\rho = 0.9$, or $C = D$ and $\rho = 0.5$, or $C = D/10$ and $\rho = 0.9$. For the scenarios satisfying one of the first three alternatives there is famine for the mean values, and the scenarios satisfying $C = D/10$ and $\rho = 0.9$ the mean values are in the middle case with positive advantage to collaborators.

Clearly, if the system will not work for an ideal reputation system that has perfect identification of free-riders, it should not work for a weaker reputation scheme. Our autonomous reputation scheme does not in general give perfect identification of free riders, so for this scheme we predict that the system will work for a subset of the 36 sets of parameter values identified above.

C.5 Simulations

We now turn to simulations for the design parameters above. The simulator simulates some aspects of a real implementation (specifically, the fluctuations over time of amounts donated and requested) that are ignored by the analytical model. However it is not a total P2P system simulator, since for example it does not deal with topology issues. We aim to investigate the effectiveness of our autonomous reputation scheme in providing a positive advantage to collaborators in the scenarios for which the analysis of the last section predicts that it is possible for a reputation scheme to do so. In order to provide a reference system, we simulated an ideal reputation scheme that perfectly identifies all free-riders.

In our simulations, the timeline is in turns, and at each turn each peer has an independent probability ρ of being in consuming state. We ran the scenarios described in Subsection C.4.2 with the value of donation $D = 10$ for all peers, using our autonomous scheme and using the

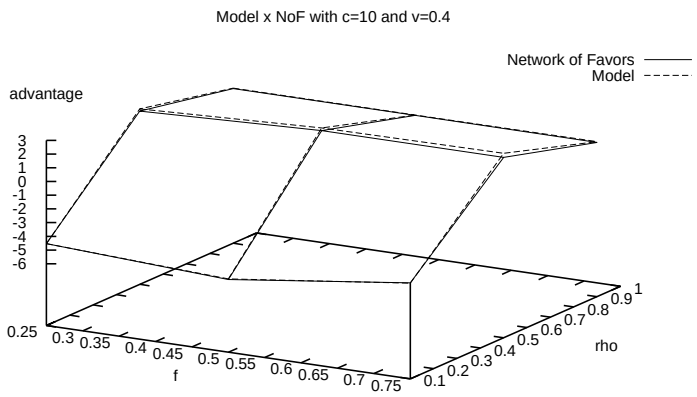


Figure C.1: Advantage to collaborators in the scenarios where $C = D$ and $v = 0.4$, varying ρ and f.

ideal reputation scheme.

For both reputation schemes the advantage to collaborators in the simulations was positive for 35 of the 36 scenarios the analysis had predicted it would be positive. Also, for 29 of these 35 scenarios the behavior of the the autonomous sceme in the simulations was close to the behavior of the ideal reputation scheme. Figure C.1 illustrates the comparison of the advantage for collaborators between a system using the autonomous reputation sceme and the ideal reputation scheme for some of these scenarios, where the eagerness level $C = D$ and the cost of donation $v = 0.4$.

The scenarios where the difference between our autonomous reputation scheme and the ideal reputation scheme was signifi cant were all the scenarios in which $C = 9D$ and $\rho = 0.1$. This difference is illustrated in Figure C.2. The scenarios in which $C = 9D$ and $\rho = 0.1$ are on the border between the famine and the middle case, so the statistical fluctuations in x_d have a greater impact on them. Indeed, it was in these scenarios that the advantage to collaborators found in the simulation using the ideal reputation scheme differed most from the the values predicted by our analysis, and also differed most from the values found in the simulation of our autonomous reputation scheme.

The sole scenario in which the system did not work using our autonomous reputation scheme when the analysis predicted that it would using a perfect reputation scheme was the one with $C = 9D$, $\rho = 0.1$, $f = 0.25$ and $v = 0.4$. This scenario is also in the border

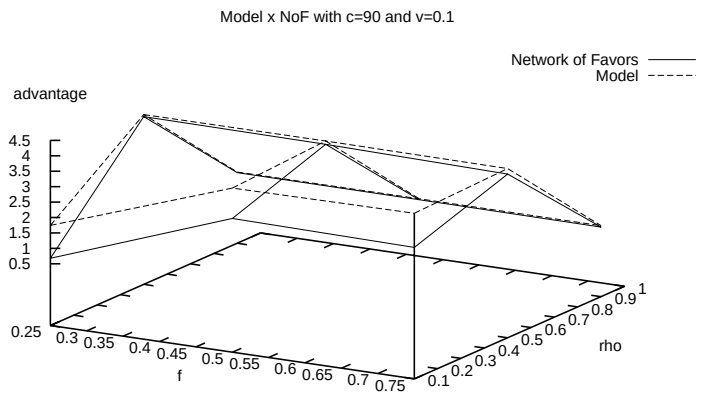


Figure C.2: Advantage to collaborators in the scenarios where $C = 9D$ and $v = 0.1$, varying ρ and f.

between the famine and the middle case, and setting $f = 0.25$ and $v = 0.4$ is enough to give a negative advantage to collaborators under our autonomous scheme. Note that these two parameter values define a scenario where there is a relatively large cost of donating resources, and few free-riders to share the resources they manage to get. The ideal reputation scheme, however, did not perform much better in the simulation of this scenario: its advantage for collaborators stays fluctuating around zero when the system reaches steady state. So, according to our definition, even when using a reputation scheme that perfectly identifies free-riders, the system did not work all the time in this scenario.

As a second step, we introduced some new scenarios where peers do not have the same D (and, hence, not the same C) or the same ρ. We investigated the cases where either D or ρ is given by the uniform distributions $U(1, 19)$ or $U(0.1, 0.9)$, respectively.

When D was given by a uniform distribution with mean 10 there was the same overall behavior as when D was set equal to 10 for all peers, and making ρ different for different peers had only a slightly greater impact. Although the mean value of ρ was equal to 0.5 in all our scenarios, the difference between the performance of the system in providing incentives for collaborations using our autonomous reputation scheme and using an ideal reputation scheme was greater in the scenarios where different peers had different values of ρ. More specifically, the statistical fluctuations in x_d made the difference in performance greater for the scenarios where $C = D$ and $f = 0.25$ and made the system not work when using our

autonomous reputation scheme in the two scenarios where $C = D$, $v = 0.4$, $\rho = U(0.1, 0.9)$ and $f \in \{0.25, 0.5\}$. Once again, these scenarios are on the border between the famine and middle cases. The statistical fluctuations in x_d arising from the differing values of ρ regularly pushed the system into the middle case, in which our autonomous reputation scheme is less efficient at rewarding collaborators. When combined with the high cost of donation $v = 0.4$, the effect was that the advantage to collaborators was negative for our autonomous reputation scheme in these scenarios.

Still, the system using our autonomous reputation scheme performed similarly to one using an ideal reputation scheme in almost all scenarios of our experiment where we predicted that any reputation had a chance of being effective (including 35 out of our original 36 scenarios with fixed ρ and D). With the exception of three scenarios where our autonomous reputation scheme proved to have a slightly different tolerance for non-contention of resources, it made the system work whenever an ideal reputation scheme would. Moreover, for the great majority of scenarios, there was only a very small difference between the measured advantage for collaborators in a system using our autonomous reputation scheme and in a system using the ideal reputation scheme.

This shows, at least for the scenarios that we simulated, that in most of the cases where it is possible to use an ideal reputation scheme, it is also possible to use our autonomous scheme without a great loss in performance. With the exception of the three border scenarios with large donation costs, although it sometimes did not perform as well as the ideal reputation scheme, our autonomous scheme still managed to provide a positive advantage to collaborators whenever the ideal reputation scheme did so. We had imagined that more complex centralized reputation schemes with global assessments of reputation would give a significantly greater advantage to collaborators than our autonomous reputation scheme, but this appears not to be the case.

Note that our comparisons between the two reputation schemes were all made after the system using our autonomous reputation scheme had reached a steady state. A system using our autonomous scheme requires some time to reach a steady state in which it has an accurate identification of free-riders. As a result, before the system reaches this state the reputation scheme should have less effect than the perfect reputation system in discouraging free-riding. We now investigate whether the eagerness level C has an impact on the time needed for the

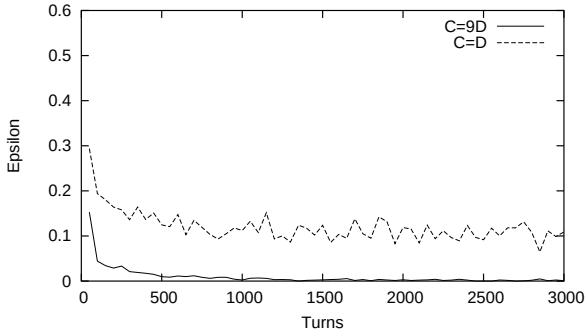


Figure C.3: The proportion of resources donated to free-riders using our autonomous reputation scheme for $f = 0.5$, $\rho = 0.5$ and different values of C

system to reach the steady state.

Figure C.3 shows the proportion of the available resources that were donated to free-riders in the last 50 turns, which we denote ϵ. When the system is in famine, ϵ expresses how well the community has identified the free-riders. We found that C does not impact on the time needed for reaching the steady state, but on the actual value of ϵ that the system shows when in steady state. We found that, except for the scenarios where $\rho = 0.9$, ϵ is approximately inversely proportional to C. When $\rho = 0.9$, although ϵ is greater for $C = D/10$ than for $C = D$ and $C = 9D$, it is very similar in scenarios where peers have one of these two last eagerness levels. We suspect that in practice, when they have one of these two eagerness levels, the consumers act as eager consumers for the system. Thus, our observations show that eagerness makes it easier for our autonomous reputation scheme to identify free-riders, although we have already found that a high level of eagerness is not a necessary condition for the system to work.

C.6 Analysis of dynamic system behaviour

Now we consider the effect of allowing peers to change their strategies. The value of f_t will vary over time according to strategy choices, whereas the values of all other system parameters are fixed over time. We assume that peers change their strategy in their own best interest, choosing to be either a free-rider or a collaborator so as to maximize their expected utility. So the gradient of f_t is positive at t if the advantage to collaborators is negative at

time t, and negative if the advantage to collaborators is positive at time t.

We do not need to offer peers a third option of leaving the system, because free-riders always have an expected utility at least as great as they would have outside the system. We assume that the choice of strategy is binary, that is, peers either choose to be a collaborator and offer all their spare resources to the community, or to be a free-rider and offer none: we do not consider the option of peers offering some but not all of their available resources.

We give a general analysis under the assumption that the values of C and D do not vary widely between peers. If this is not the case, then the system is harder to analyse in general, but we describe one example with heterogeneous peers for which it is possible to determine the system's dynamic behaviour.

C.6.1 Analysis of dynamic behaviour

For this subsection we assume that the values of C and D do not vary widely between peers. More specifically, we assume (as in Subsection C.4.2) that we can determine whether the system is in famine, glut or the middle case for the system at time t, and whether the advantage to consumers is positive or negative, by calculating the case and the sign of the advantage to consumers for the expected values of x_d, x_c and x_f at that time. In practice statistical fluctuations may temporarily move the system into another case, but we assume that these excursions are sufficiently rare and short-lived that they can be ignored when we are considering the large-scale long-term dynamic behaviour of the system.

Figure C.4 illustrates the dynamics of the system given these assumptions. The ratio of the mean values for x_d and x_c is independent of f_t, so is fixed over time. Therefore if famine holds for these mean values for the initial state of the system, it continues to hold for the subsequent evolution of the system. As free-riders become collaborators in a system with a famine of donations, the average amount of resources donated to the system at a given time increases, but the average amount consumed at that time also increases, because the new collaborators are donated (and hence consume) more resources than when they were free-riders. This increase in consumption is large enough to keep the system in famine. So if the system is initially in famine it should remain in famine as f_t decreases, (except for rare excursions given by statistical fluctuations), and eventually all the free-riders become collaborators.

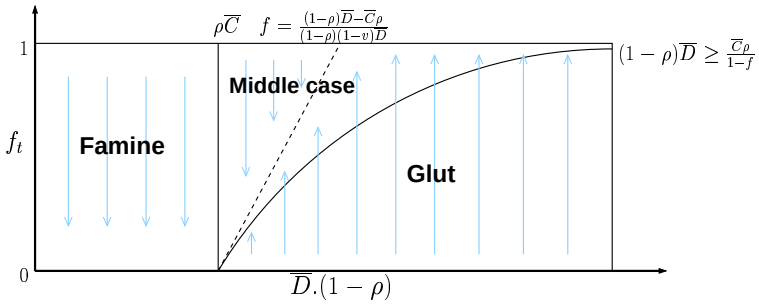


Figure C.4: Dynamics of the system, varying f_t and $\overline{D}.(1-\rho)$.

If the system is initially in glut then f_t will increase until eventually the system is no longer in glut - at this point it will be in the middle case. At the border between the glut and middle cases free-riders have a higher expected utility than collaborators.

For the middle case, either the advantage to collaborators is negative for all $f_t \in (0,1)$, in which case collaborators will eventually die out, at which point the system will not work, or else it is a monotonically increasing function of f_t for $f_t \in (0,1)$ which tends to $-\infty$ as $f_t \to 0$ and to a positive number as $f_t \to 1$. For this second alternative the system will evolve to a stable equilibrium at which $f_t > 0$ and free-riders and collaborators have equal expected utilities. At the stable equilibrium the system does not work, because the advantage to collaborators is zero. In practice the system may oscillate around the stable equilibrium rather than reaching it precisely, because of the statistical fluctuations in the value of resources donated and requested; but the average over time of the advantage to collaborators for the oscillating system will still be zero.

It follows that if there is not a famine of donations, the proportion of free-riders will evolve over time to a value at which the system does not work. It is possible that the system initially is in a state in which it works, if the conditions for the middle case hold and the advantage to collaborators is positive, but it will eventually evolve to an equilibrium state in which the system does not work. This happens even though in this analysis we are assuming perfect identification of free-riders. On the other hand, if there is a famine of donations, then the system will work and free-riders will eventually die out.

This gives a relatively simple heuristic for checking if the system is eager enough for a reputation system to have a chance of acting to drive out free-riding by prioritizing donations to peers with high reputation: assuming that all peers choose to free-ride or not so as to

maximize their expected utility, free-riding can only be driven out if there is a famine of donations, ie.

$$\overline{D}.(1-\rho) \leq \overline{C}.\rho \tag{C.2}$$

In our simulations, peers did not change strategies. However for most of the scenarios we simulated, after some time interval the effect of the autonomous reputation scheme was similar to that of a scheme with perfect prioritization, and this held for a range of values of f. It is therefore reasonable to expect the eventual behaviour of a community under our autonomous reputation scheme to be similar to that predicted by this analysis, provided that the community has enough time to identify free-riders when peers change strategy.

C.6.2 An example with heterogeneous peers

In this paper we have assumed that the values of C and D do not vary very widely between different peers. In particular we have assumed that the standard distributions of these values are small and that it is sensible to discuss their means. For some potential application systems this assumption is unrealistic. If the values do vary widely our previous analysis will not apply, although it may still be possible to predict the system behaviour.

For example, begin with a network in which the values of C and D do not vary widely between peers, and the average amount of resources donated in a given timeslot is less than half than the amount requested. Clearly, there is a famine of donations. Now make the set of peers heterogeneous by adding a new peer that is in donating state at least half the time, and has such a high value of D that when it is in this state it is able to meet all the unfulfi lled requests in the system. The new peer's expected utility is greater outside the system than as a contributor within the system, even if all the other peers donate all their free resources to the new peer when it is in consuming state. So the new peer will become a free-rider. The remaining peers will eventually detect that this has happened, and after that the system will evolve as in our analysis, except that there is one additional free-rider (the new peer): eventually, all other peers will become contributors.

Essentially, our reputation scheme encourages the donation of resources by a peer only if there is a reasonable chance of the donation being repaid.

For distributions of C, D with large deviations which are not as extreme as in this example (for example, Zipf distributions, which we have investigated), the system behaviour is hard to analyse, as it is contingent on the behaviour of peers with high values of C and D. More research needs to be done.

C.7 Conclusion

In this paper we have demonstrated through simulations that in a network of similar peers, our autonomous reputation scheme is sufficient to discourage free-riding when there is a famine of donations. Our analysis of a system model indicates that when there is not a famine of donations, no reputation scheme should be able to discourage free-riding by prioritizing donations to peers with high reputations.

In our simulations of the autonomous reputation scheme for sample scenarios in which there was a famine of donations, there was an incentive for collaborating as opposed to being a free-rider in almost all scenarios where an ideal reputation scheme would also provide such an incentive. The cases where the results where different showed that our autonomous reputation scheme has a slightly worse prioritization than the ideal reputation scheme and therefore requires slightly more contention for resources to keep the utility of free-riders low. For the majority of the scenarios, both schemes performed similarly. The autonomous scheme discourages free-riding by prioritizing donations almost as well as the ideal reputation scheme, despite being very lightweight and easy to implement, and requiring neither central coordination nor a cryptographic infrastructure.