Webservice Protocol Design for Economic Liberty and Observability

Norbert Bollow
Bollow Software Economics Research
Weidlistrasse 18
CH-8624 Grüt
nb@norbert.ch

ABSTRACT

One big potential benefit of the webservices paradigm is in reducing the costs of inter-firm business transactions. That should allow small and medium-sized enterprises to compete successfully with big firms. This paper considers specifically the economic needs of *peer-to-peer business alliances,* defined as multiparty business alliances which are not under the control of any single firm or any small group of alliance members, so that each participating firm has full economic liberty. This organisational form is appropriate for example for Free Software businesses. The main conclusions are that achieving *economic observability of business transactions* is of great importance, and that this is difficult to achieve with the Remote Procedure Calls paradigm of JINI or XML / HTTP / SOAP based webservices. The problem can be overcome by using the SXDF / QQP / QRPC suite of webservice protocols, which provides the needed flexibility, as well as performance improvements for more standard webservice needs.

Keywords

Webservices, Protocol Design, Observability of Business Transactions, Peer-to-Peer Business Alliances, Free Software Business, Fair Trade, Small and Medium-sized Enterprizes, World Market Access, Philathropic Business Alliances.

1.INTRODUCTION

When the webservices paradigm initially emerged, there was a widespread expectation that it would lead to a profound, fundamental paradigm shift with strong, disruptive effects on the entire realm of information technology relegated commerce. This was expressed with words like “a world driven by web services is on its way, and it would be fair to say that only the prepared will survive”, and for example Microsoft Corporation acted on this belief by “betting the company” on the “.NET” strategy [Gov01a].

Today webservice protocols are in widespread use, but they have not had the expected disruptive economic impact. The viewpoint of the present paper is that the main cause for this is a profound misunderstanding of the future webservice-powered economy as a battleground for dinosaur-like corporations of which “only the prepared will survive”, and that this misguided expectation has led to webservice protocols with serious shortcomings which effectively prevent the webservices paradigm from achieving its full economic potential.

Economically, the main expected benefit of webservice technology is in the area of reducing transactions costs, both in the area of *internal transaction costs* for interactions between different software systems within the same company, as well as *inter-firm transaction costs* of negotiating and executing business transactions with other companies. Now relatively high costs of business transactions between firms are the main reason for the existence of big companies, see [Coa37a]. When the successful introduction of a webservices paradigm for inter-firm business transactions reduces these transaction costs, the result will be that a business alliance of many small companies which collaborate in the areas of strategy and marketing will have an economic advantage in comparison to a single big company. (Collaboration in the area of marketing can be implemented for example through “word-of-mouth" strategies as described in [Mis99a].)

The discussion of the economics of such a business alliance in section 2 of this paper leads in several ways to the conclusion that it is important for the webservice protocol framework to provide software

tools for appropriately observing webservice transactions. This notion of "observability" has been inspired by how the notion of measurable quantities has turned out to be fundamental for modern physics: The same physical phenomena which allow a quantity to be measured are what makes it relevant as part of the physical world in the first place. Hence the fundamental theories of physics should not involve anything which, for reasons of principle, is impossible to measure. This realization has led to the development of very important and fruitful theories like quantum mechanics and the theory of relativity.

The main technical content of this paper is in section 3 which proposes a suite of webservice protocols that is designed to help with providing this *observability* which in the previous section was shown to be important. By contrast, this is difficult to achieve with the currently widely-used flavors of webservices, which are on one hand Sun's JINI [Wal99a] and on the other hand the XML / HTTP / SOAP based webservices paradigm recommended by World Wide Web Consortium [Box00a].

Finally section 4 discusses Free Software businesses and firms with "fair trade" concerns as examples of categories of businesses where the idea of forming peer-to-peer business alliances as proposed in this paper can be expected to lead to significant economic benefits.

2.BUSINESS ALLIANCE ECONOMICS

In this section we discuss the economics of a business alliance, which is assumed to be in a stage of "preliminary planning" so that we are free to design software systems in any way which will allow the business alliance to function efficiently, without worrying about the time requirements for implementing a new webservice protocol suite if that turns out to be desirable.

Furthermore we assume that the member companies of the business alliance share a common interest, for example the alliance could consist of businesses which are committed to pursuing a specified humanitarian agenda as a secondary purpose besides the primary purpose of every business which is earning money. Examples of common interests which could provide a strong cohesive force in a business alliance include e.g. the agenda of the Free Software movement, or the worldwide fight against poverty, or the desire of Christians to do what is pleasing to God. Without the cohesive effect of such a common interest, business alliances of many small companies are likely to be so unstable that the business alliance will be unlikely to ever grow big enough to be able to seriously compete with a big company. Another example of how a common interest can facilitate a business alliance is the "open source" software development process, which (from the economic perspective at least) is a form of business alliance of all contributors.

2.1 Markets

The fundamental assumption of this paper is that it is an important aspect of the business alliance under consideration to facilitate trades of some kinds of economic goods among alliance members. Here the term *economic good* is used for anything that can be the object of an economic transaction. Examples of economic goods include books, houses, cars, libraries, the right to use a given library, computer programs, recorded music, internet connectivity, and services of all kinds.

An economic good is called a *private good* if it is possible to limit access to the good (the good is *excludable)* and if there is a limit to how much it can be used (the good is *depletable*).

For example, computer hardware, all kinds of machines and all kinds of services are examples of private goods. Information and ideas are in a fundamentally different category of economic goods because they are not depletable. Patent laws make some or many categories of ideas excludable.

In *markets for private goods*, prices play an extremely important role, by giving appropriate incentives both to producers and consumers of goods:

- An incentive for consumers of goods to use scarce resources efficiently.
- An incentive for producers of goods to meet the actual desires of consumers.

In the communist, planned economies of Eastern Europe, the experiment has been tried for several decades to run an economy without allowing the prices of private goods to be at an appropriate level for having this function of providing appropriate incentives to both producers and consumers. Even though these countries tried very hard to economically outperform the capitalistic countries, they failed to even keep up with the "West". However it is clear that this incentive system can function only to the extent that buyers and sellers are actually aware of the business transaction and the corresponding price. This economic importance of observability is independent of whether the business transactions are negotiated personally between humans or whether they're conducted in an automated manner by means of software agents and webservice protocols.

2.2 Business Relationships

Consider for a moment the category of economic systems where buying decisions are not influenced by the vendor's good or bad reputation. In such systems, auctions are a generally desirable method for conducting business, because with auctions, the allocation of scarce resources is efficient and fair, even if only a single item is traded.

By contrast, whenever reputation and trust are important, it is preferable to conduct most business through long-term business relationships, using auctions only as a last resort. The reason is that (assuming a genuinely competitive market) you pay the same market price regardless of whether you buy via auctions or through a long-term business relationship. However, buying via auctions increases risk, because some of the purchases will be from less trustworthy vendors, and this increased risk needs to be considered as a cost for the buyer, a cost that however does not result in an increased revenue for the seller. This explains why in the "real-world" economic system, auctions are used only under special circumstances, and most commercial transactions are conducted through long-term business relationships.

One function of a business alliance is to build trust among the members, this has the effect of decreasing the cost of establishing business relationships between the members. However this can work in a sustainable way only if the behavior of the members of the business alliance is observable at least retrospectively with respect to all the standards of behavior upon which the business alliance is founded. Here the term *retrospectively observable* means that it should be possible to verify some time after the event whether one of the principles of the business alliance has been violated, even if allowing to check this immediately might be an unacceptable for legitimate privacy reasons.

2.3 Peer-to-Peer Business Alliances

A *network* (in the technical sense) is a communication system which allows a set of *nodes* (that may change over time) to interact with each other. The term *peer-to-peer network* is used for networks which may have centralized as well as decentralized aspects, but where the decentralized aspects are considered to be more important.

The kind of business alliance which we discuss in this section is an economic analogue to a peer-to-peer network. The nodes are the participating businesses, with connections between the nodes corresponding to business relationships.

DEFINITION: A ***peer-to-peer business alliance*** (**P2PBA**) is defined as a multiparty business alliance in which all decision-making processes are at least potentially totally decentralized.

A P2PBA can be started in such a way that it has simple, centralized administrative and decision-making processes initially, provided there is a clear, legally-binding commitment that whenever one or more alliance members may desire to decentralize some of the administrative and decision-making processes, they will always have the freedom to separate themselves from part or all of these centralized administrative and decision-making processes without losing the benefits of business alliance membership. For example, a business alliance in which the negotiations for purchases of laptop computers are centralized (the business alliance members pay a negotiator who will negotiate on their behalf with laptop vendors) can be a P2PBA if alliance members have the freedom to "opt-out" from this centralized feature of the business alliance in a way which allows them to stop contributing financially to paying the negotiator, without losing any benefits of business alliance membership that are not directly related to the negotiator's work, and if business alliance members have the freedom to establish competing bulk laptop purchase services, if desired. This *freedom to fork administrative and decision-making processes* means that P2PBAs can be considered to be an economic equivalent of Free Software.

We say that members of a business alliance are *rationally egoistic,* or acting *in their rational self-interest,* if they act in a manner which is consistent with rationally seeking to further their own economic interests, without concern for the welfare of others. (In a well-designed business alliance there are appropriate incentive systems so that whenever it is important for the business alliance as a whole that something should be done, doing that will be in the rational self-interest of at least some members.)

One crude, but frequently-used, method of approximating human behavor is to assume that everyone acts in this manner. When planning business alliances it is however helpful to consider also the following three additional categories of alliance members:

We say that a member is *unconditionally altruistic* if the member seeks to further the interests of some set of other members, or of the business alliance as a whole, even when there is no economic incentive to do so, and even when everyone else is assumed to be rationally egoistic. (There is a variety of motivations that can move people to take actions which are not economically motivated.)

Contigent consenters are members who are willing to follow rules even if cheating would be in their rationally egoistic self-interest, but who are willing to do this only as long as they are under the impression that most others do the same.

Finally, we say that a member is *anti-social* if the member seeks to harm some set of other members, or the business alliance as a whole, even when there is no economic incentive to do so.

Similar to peer-to-peer networks, business alliances should be designed to be robust against disruption attempts from a small number of anti-social members. Making this a goal implies that there should be a way to determine whether and to what extent the goal has been achieved. This again requires mechanisms for observing (and logging) the relevant aspects of webservice interactions so that it can be determined, at least in hindsight, whether and when such a disruption attempt has occurred.

One current example which shows that the possibility of anti-social action should not be neglected is the behavior of the company known as "The SCO Group": Through making totally unsubstantiated claims that commercial users of the operating system kernel known as "Linux" need to buy an expensive "SCO Intellectual Property License for Linux", and through launching multiple expensive lawsuits which "The SCO Group" cannot reasonably expect to win, the company has maneuvered itself into a position where it cannot possibly survive. (Details are available at the "Groklaw" weblog, http://groklaw.net .) Since the company's behavior makes no economic sense from the perspective of the company's business interests, it must be classified as "irrational", even when the decisions of SCO's managers may make sense from the perspective of their own personal financial self-interest.

Altruistic members on the other hand are nice to have, and the experience with peer-to-peer networks shows that there is often a high percentage of altruistic nodes among the first few nodes that join together to form a peer-to-peer network. Peer-to-peer networks however need to be designed with appropriate incentive structures so that once the network is well-established, it will continue to function properly even if there are no altruistic nodes.

According to [Feh00a], about 25% of people act egoistically, and a small minority are altruists, while the vast majority are contingent consenters. This implies that business alliances are well-advised to carefully monitor the rate of cheating attempts which are found out, and if this starts to increase because some of the contigent consenters are starting to cheat, punishments for cheating must urgently be made more severe, more frequent (this may require allocating more resources for rules enforcement) or more publicized.

2.4 Evolving Standards

Humans are very creative in inventing new ways of doing things which increase their personal profit. This is very good as long as the net total effect of these economic activities is positive. Unfortunately that is not always the case. A very clear example of this problem is the phenomenon of email spam, which causes huge damage for the economy as a whole, not only by wasting time of internet service providers and email users, but also by decreasing the overall value of the email system through automated "spam filter" programs which sometimes reject (or even silently discard) also legitimate email messages, and through the frustration of "end users" who stop using the email system effectively when their mailboxes are overflowing with worthless spam messages. For this reason, business alliances are defined not only by how the members of the alliance are empowered to interact, but also by the standards of behavior which are established for the purpose of protecting the alliance from egoistical actions which on the whole do more harm than good.

From the perspective of someone who wishes to create a highly successful business alliance of a significant number of small and medium-sized companies, this aspect of how the business alliance can grow without losing its integrity is one of the most important considerations. During the initial stages of a business alliance, when all members know each other well, antisocial selfish behavior is very unlikely, so that no formal rules are needed for discouraging it. However when the business alliance grows some standards need to be introduced which allow to make the distinction between acceptable social and commercial activities on one hand, and conduct which like spamming is not acceptable on the other hand.

At least in P2PBAs, compliance with standards of behavior needs to be verifiable in a fully automated manner, because otherwise the decentralized introduction of new and modified standards of conduct will not be practically feasible. This implies that technical infrastructure needs to be available which business alliance members can use to collect data about their commercial activities which they can use to prove that their commercial conduct has been blameless. These software system need to be designed with great flexibility because the precise standards of behavior which will turn out to be important are not known in advance, and it also

needs to be designed with the goal of protecting legitimate privacy interests, so that for example no business alliance member will be unable to prove the righteousness of its conduct without disclosing data about its customers or other essential business secrets.

2.5 The Problem of Cheap Identities

Membership in a business alliance as proposed here should be achievable only after a "membership commission" has made inquiries about the background of the prospective new member which are sufficiently careful to prevent people whose companies have been thrown out of the business alliance for violating its rules from simply rejoining the alliance with a new business name.

This avoids the problem of "cheap identities" which in the context of the email system greatly increases the difficulty of taking effective action against spam.

If it is easy to obtain a new "identity", that can be seen as a violation of the principle of observability, because it becomes difficult or impossible to properly observe the various actions of an economic agent when they are conducted under many different different identities, which are not connected with each other in any obvious way.

2.6 Firewall Considerations

When webservice protocols are used for business transactions between different companies, that usually involves exchanging confidential data over insecure communication networks such as the internet, and it also involves granting some degree of access privileges to one's business partners.

These two aspects of digital business transactions impose two requirements on the technical implementation which are close to conflicting with each other. One one hand, corporate firewalls need to be able to examine incoming webservice requests and response data to prevent abuse of the webservice interfaces which business partners want to expose to each other. On the other hand, malicious outsiders must be prevented from being able to extract any valuable information from observing the exact same data stream which the firewall uses to determine whether a given chunk of data purporting to come from a business partner should be accepted or rejected.

Fortunately these requirements are not truly in conflict with each other, as a securely encrypted SSL connection can be established between the corporate firewalls of the business partners. This argument shows only that it is not appropriate to use the security model of the HTTPS protocol directly between the webservice server and the client computer. Rather the correct way of using the XML / HTTP / SOAP suite of webservice protocols is to use HTTP between the client computer and the corporate firewall on the client's side, and again HTTP between the corporate firewall on the server's side and the webservice server itself, while HTTPS is used on the insecure network between the two firewalls. This means that HTTP → HTTPS and HTTPS → HTTP gateways need to run on both firewalls.

3.THE SXDF / QQP / QRPC SUITE OF WEBSERVICE PROTOCOLS

This section discusses conclusions of the above considerations for the webservices system of a P2PBA. To some degree the arguments of this section can also be made for CABAs (based mainly on the difficulty of gradually updating centralized systems in view of evolving requirements), but for the sake of simplicity of the arguments, and for the sake of making it possible to achieve a conclusive result, this section assumes that the business alliance under consideration is a P2PBA. Then there are several significant reasons against basing the system on the widely popularized webservice protocol suite which consists of HTTP or HTTPS for data transport, XML as extensible data markup scheme and SOAP for the actual webservice request and response layer.

First of all, process of recording all the essential data which is important for the observability discussed in the previous section will happen in practice only if it does not slow down or otherwise interfere in unpleasant ways with the business transactions themselves. In this context, any plan to use a pure RPC (Remote Procedure Call) paradigm, on which the combination of HTTP and SOAP is based, must appear unwise, because for RPC protocols the requirements are in conflict with each other that on one hand the process of recording essential data about business transactions must be reliable, but on the other hand this process must not create delays in the business processes which are implemented by the computer systems which make the webservice requests that should cause this data to be recorded. This problem is solved by the Queueable Remote Procedure Protocol discussed below.

QRPC also adds flexibility to the webservice system similar to how in UNIX operating systems, great flexibility results from the possibilities of redirecting the standard input, standard output and standard error streams, and combining commands into "pipelines". Because of the reasons discussed in point 2.4 above, it is important for P2PBAs to have a webservices system with as much flexibility as possible. The basic building blocks for this

flexibility are the Quick Queues Protocol (QQP) for data transport and the Simple Extensible data Format (SXDF) which shares XML's good properties of basic human readability and interoperability with extensions, but which allow for much more efficient, less resource intensive, parsing. This is important for example for allowing firewalls to examine the legitimacy of data which is allowed to pass through the firewall without turning the firewalls into a point of congestion for all incoming and outgoing data traffic.

3.1 Simple Extensible Data Format

Over the past few years, the Extensible Markup Language (XML) [Bra00a] has become a widely used method for data markup. The Simple Extensible Data Format (SXDF), pronounced “sixdaf”, aims to combine the good properties of XML with a simple syntax which can be parsed efficiently by computer programs.

SXDF shares the following important properties of XML:

- It is a universal data format which can be used for expressing arbitrarily complex data.
- It is a text-based format, which makes it convenient to debug protocol interactions which use the data format.
- Data can be validated in an automated manner to ensure that it adheres to a specified data structure.
- There is great flexibility in how the data format used by a given protocol can be extended without breaking existing implementations of the protocol.

SXDF differs from XML in that with SXDF the main design goals are simplicity, and allowing efficient parsing by computer programs.

SXDF is not a "markup language". It is not intended for data which will be edited with a text editor.

A sequence of bytes (eight-bit octets) which satisfies the requirements of the SXDF specification [Bol04a] is called a "SXDF resource". Here is an example:

```
484:// here is some data in SXDF format
1%
 8:Booklist=3@
  5%
   5:Title=16:Hardware Hacking
   6:Author=19:Kevin Mitnick (Ed.)
   4:Year=4:2004
   4:ISBN=13:1-932-26683-6
   9:Publisher=8:Syngress
  5%
   5:Title=12:We the Media
   6:Author=11:Dan Gillmor
   4:Year=4:2004
   4:ISBN=13:0-596-00733-7
   9:Publisher=8:O'Reilly
  5%
   5:Title=22:Matrix Decision Making
   6:Author=21:Alex Lowy & Phil Hood
   4:Year=4:2004
   4:ISBN=13:0-787-97292-4
   9:Publisher=11:Jossey-Bass
;
```

Here is the same data expressed in XML format:

```
<!-- here is the same data expressed
     in XML markup -->
<Booklist>
 <Book>
  <Title>Hardware Hacking</title>
  <Author>Kevin Mitnick (Ed.)</author>
  <Year>2004</year>
  <ISBN>1-932-26683-6</ISBN>
  <Publisher>Syngress</publisher>
 </Book>
 <Book>
  <Title>We the Media</title>
  <Author>Dan Gillmor</author>
  <Year>2004</year>
  <ISBN>0-596-00733-7</ISBN>
  <Publisher>O'Reilly</publisher>
 </Book>
 <Book>
  <title>Matrix Decision
         Making</title>
  <author>Alex Lowy &amp;amp;
          Phil Hood</author>
  <year>2004</year>
  <ISBN>0-787-97292-4</ISBN>
  <publisher>Jossey-Bass</publisher>
 </Book>
</Booklist>
```

A key feature of the SXDF data format, inspired by [Ber97a], is that all strings, lists and dictionaries which are embedded in a SXDF resource are preceded by an integer number which gives the number of bytes in the string, the number of items in the list, or the number of key-value-pairs in the dictionary, with the result that parsers for the SXDF format will generally be less complicated and faster than parsers for the XML format. In addition, preceding each string value with an integer which indicates its length has the benefit that string values can contain arbitrary binary data, without any need to escape special characters or to use base64 encoding as required by XML. For some purposes, such as e.g. video streaming, this alone gives SXDF an essential advantage over XML.

A SXDF resource may contain a `DSD` element, which, if present, is an assertion that the SXDF resource confirms to a particular SXDF Data

Structure Description (DSD). The value of this DSD element is either a string which references the DSD by means of an URL, or a dictionary which contains the DSD explicitly.

A DSD consists of a dictionary in which one key is "resource" and the other keys are the various types of elements in the data format which is to be described by the DSD. The values corresponding to these keys describe the various allowed kinds of data, in a rough manner which can be verified easily.

Here is a SXDF Data Structure Description for the above Booklist example:

```
236:6%
 8:resource=1%
  8:Booklist=12:1*1@8:Booklist
 8:Booklist=6:*@4:Book
 4:Book=5%
  5:Title=5:1*1@1:t
  6:Author=5:1*1@1:t
  4:Year=6:1*1@2:t4
  4:ISBN=7:1*1@3:t13
  9:Publisher=5:1*1@1:t
 1:t=2:*t
 2:t4=4:4*4t
 3:t13=6:13*13t
;
```

This is read as follows: The entire SXDF resource (which happens to contain a DSD) consists of 220 bytes. It is a dictionary with six entries, each of which define a named data type. The last three of these are easiest to understand: `t4` is defined to refer to a string which contains exactly four bytes of textual data, while `t13` is defined to refer to a string of exactly 13 bytes of textual data. (A string of not less than 4 and not more than 13 bytes would be specified as `4*13t` if it is restricted to textual data, or as `4*13b` if it may contain arbitrary binary data.) All counts are in terms of bytes, not characters, to make it is easy to parse and verify SXDF resources without need to consider the possibility that the number of bytes per character can be variable. All textual data in SXDF resources can be in UTF-8 encoding or it can be a sequence of 16-bit unicode characters that starts with the unicode Byte Order Mark, `0xFE 0xFF` or `0xFF 0xFE`, depending on endianness. (The case of UTF-8 encoding can be easily and reliably distinguished from this, because in UTF-8, all bytes have values is the range `0x00-0xFD`.)

In the above DSD example, the first dictionary entry defines the type “resource” as consisting of a dictionary with a single required entry named `Booklist`, the type of the value is also named `Booklist`. This type is then defined in the next line as an array that can contain number of entries of type `Book`. The type `Book` is defined as a dictionary with five required keys `Title`, `Author`, `Year`, `ISBN` and `Publisher`.

If this DSD is published at http://SXDF.org/Booklist-DSD.sxdf, this URL can be added to the Booklist example, for purposes of validation, by means of adding

```
3:DSD=23:http://SXDF.org/Booklist-DSD.sxdf
```

Alternatively, the DSD can be included literally in the SXDF resource.

3.2 QQP - Quick Queues Protocol

The fundamental idea of "webservices" is to access functionality on another computer by means of standardized multi-purpose protocols and a standardized extensible data format. For example, when using the "webservices" paradigm to specify a message transport system, you don't specify the protocol and data format from scratch like it is done for SMTP in [Kle01a] and [Res01a]; instead you specify them in terms of a general-purpose data format (often XML) and a general-purpose protocol for transporting data between computers.

QQP, the Quick Queues Protocol, is such a general data transport protocol, specifically for data in the SXDF format (see section 3.1 above). The QQP specification [Bol04b] describes an efficient way to transmit a stream of data over as TCP or SSL or similar data connection. This data stream consists of a sequence of SXDF resources contains an Element named `Action` which specifies what the receiver should do with the resource.

When the data connection is first established, the Receiver transmits a special, initial SXDF resource called the “Greeting”, which contains some information about the receiver which may be useful for optimizing efficiency of the data transfer, and also a field named `Capabilities` which contains a list of string values like for example

```
12:Capabilities=4@
 4:qrpc
 12:sign-forward
 4:http
 4:soap
```

The elements in this list of capabilities correspond to what URI types are possible as values of the `Action` field. For example, `qrpc://` URIs are used for the QRPC protocol (see section 3.3), for webservice execution requests, related “signals”, as well as additional data which is sent in support of such requests, or data which is sent in response to such requests. A `sign-forward://` URI indicates a request to digitally sign some data and then forward it to a specified recipient; this kind of request is important for some types of anti-spam systems. A `http://` URI indicates that the QQP Receiver should act as a QQP→HTTP gateway

(which can interact with CGI programs via HTTP POST and/or HTTP GET), and a `soap://` URI indicates that it should act as a QQP→SOAP gateway.

Many QQP Receivers will provide a subset of the capabilities of the above examples. The mechanism is extensible, but due to the great flexibility of the QRPC protocol, extending it through defining semantics for additional URIs will typically be of interest only for purposes of adding gateways to additional legacy protocols.

After the sender has opened a TCP connection to port 26 or an SSL connection to port 27, and after the sender has received the "Greeting", it can either close the connection (this may be appropriate if the Greeting indicates that the receiver's buffers are already almost full) or it can send a SXDF resource (which must not be larger than a limit which the receiver has indicated in the Greeting). When this SXDF resource has been received, the Receiver sends a status string. The first character of this status string is a digit in the range from 1 to 6 with the following semantics: 1 means “redundant”, the SXDF resource is being ignored because a SXDF resource with identical `ResourceID` has been received before. 2 means “Received alright; next resource please”. 3 means “Received alright; closing connection”. 4 means “Temporary error; please try again later”. 5 means “Permanent error: something is wrong with this SXDF resource”. 6 means “Undefined status: something is wrong, and it isn't even known whether this condition is temporary or permanent.”

If this Status Code (see below) is 3 or greater, the receiver closes the connection. Otherwise, the sender is free to either close the connection or send another data item.

One important aspect of QQP is that the Receiver may check for resource duplication and/or for errors in a way which for implementation-related reasons does not allow to generate the "1" or "5" status code during the QQP protocol interaction. Whether such errors should be reported and how they should be reported is controlled by an optional element in the data resources named `ExceptionsTo`. If this is present, the receiver reports such errors by sending a QRPC Exception resource to the URL specified as the value of the ExceptionsTo URL of the data item for which the error has been detected; if the data item has no `ExceptionsTo` URL, Receiver will not generate a QRPC Exception resource.

3.3 Queueable Remote Procedure Calls

QRPC is a webservice protocol designed to achieve greater flexibility and performance improvements over what is possible with synchronous XML-based protocols such as XML-RPC or SOAP. QRPC uses SXDF instead of XML as its basic data format, and the data is typically transported by means of the Quick Queues Protocol (QQP).

Unlike SOAP and XML-RPC, the QRPC protocol is also designed to support stateful webservices. Four types of SXDF resources are used to implement the QRPC protocol, which are named ExecutionRequest, StreamedData, Exception and Signal.

The client creates a “webservice session” by means of an ExecutionRequest. The ExecutionRequest will generally contain some parameters. Optionally, additional data may be transmitted as part of the same webservice session by sending StreamedData resources in addition to the ExecutionRequest, until one of the resources contains an EOT ("end of transmission") element which effectively closes the data stream from the requestor to the webservice server. (When no StreamedData resources will be sent, the EOT element will already be included in the ExecutionRequest).

The client waits before sending the next data packet until a response packet has been received to the ExecutionRequest, so that the client knows that the ExecutionRequest has been accepted. Afterwards the client may send multiple data packages without waiting for response packets. Some webservices will generate a response packet for each data packet which is received from the client, while others will will receive many data packets but generate only two response packets, one after the ExecutionRequest has been accepted and a second at the end of the webservice session. The client is required to stop sending data immediately if it receives a response which contains an EOT element. (Waiting for each response is necessary if the next packet cannot be computed until the response to the previous packet has been received, but it many situations it would just needlessly slow data transmission.) If the server receives a data package with a sequence number which is higher than the expected next sequence number, it will not process this package until all packages with lower sequence numbers have been received and processed. If this is not possible (e.g. because the necessary resources are not available for buffering packages, or because a configured maximal waiting time has expired) the server generate an Exception and in addition send a StreamedData data package which contains an EOT element.

3.3.1 Exceptions

The server reacts to errors by generating an Exception. Depending on the severity of the error,

the webservice process may possibly need to be terminated, in which case in addition to the Exception a response data package which contains an EOT element is generated.

3.3.2 Redirection

The ExecutionRequest may optionally contain a qrpc: URL to which response packages should be addressed, and it may optionally contain a URL to which Exception packages should be addressed. By default both types of packages are directed to the Sender URL of the original ExecutionRequest.

3.3.3 Signals

Signals are data packages sent from the client to the server which do not contain a sequence number, and which may be sent at any time, i.e. even after the final data package (which contains the EOT element). For example, the server may implement a type of signal which causes it to immediately generates an exception, send a final response data package (containing an EOT element), and discard any remaining data from the client which may still be unprocessed.

3.3.4 Handling Exceptions

Server and webservice implementations should have detailed documentation of all exceptions that they can generate. Generally the client forwards all Exception packets which it cannot handle to a general exception handler application that will inform the system operator in an appropriate manner. If the client is not capable of handling any exceptions, redirection of Exceptions to the a general exception handler application can be used.

4.WHO WILL BENEFIT?

Since no experimentally-verified quantitative economic theory is available which would allow to adequately describe the economic effects of introducing a webservices framework as proposed, it is difficult to predict whether the disruptive effects of this technology will be small or great when seen from the perspective of the economy as a whole. Such quantitative estimations will become possible only after webservice implementations with economic observability have been used for a while, and some of the resulting data has been made available for scientific evaluation.

It appears safe however to predict that there will be a significant economic impact on groups of businesses which are already agreed in the common purpose of some kind of philanthropic concern, firms which pursue not only the primary goal of every business of earning money, but where the pursuit of some specific philanthropic goals is also an essential element of the company values.

4.1 Free Software Business

A good example of this category of firms with philanthropic goals are *Free Software* businesses where the philanthropic agenda consists in maximizing the personal and economic freedom of computer users.

How many people are there who either have a small firm in this category or who would love to start one as soon as a good business opportunity presents itself? This category of Free Software businesses does not include firms like IBM or Novell which pursue an "open source" strategy simply as an economic wager that the *collaborative software development paradigm* will allow their company to generate greater profits than it otherwise could. Among the large variety widely-used Open Source and Free Software licenses, there are a few which express the "software freedom" goals of Free Software businesses so well that everyone who makes some software available under such a license could be sonsidered a potential Free Software entrepreneur. Among these licenses, the *GNU General Public License* [Sta91a] is by far the most widely used; the main properties of this lincese are that it grants a lot of "software freedom" rights to every user and that it ensures that derivative works will always remain Free Software which everyone can freely use, modify and redistribute in original or modified form. The best available method for obtaining a rough estimate of the current total number of actual and potential Free Software business entrepreneurs may be to go to the "statistics" page of "Freshmeat.net", the leading website which tracks interesting Free Software and Open Source software packages, and take the number of Free Software projects which use this license. As of March 2, 2005, the number given is 26,386 which is about two-thirds of the total number of packages on the site. The plausible conclusion from this is that the current total number of actual and potential Free Software business entrepreneurs is probably also a five-digit number.

Since this appears to be a promising group of actual and potential entrepreneurs who may be interested in trying out the ideas of this paper, it is important to consider any foreseeable concerns which this group of people is known to have about webservice technology and business alliances. In particular, not every potential use of webservice technology is morally acceptable from the perspective of Free Software philosophy, see e.g. [Ban03a]. The webservices paradigm is viewed as good when, through making it easier to integrate technical processes and business processes between different computer systems, it has the effect of increased

economic liberty. On the other hand it is also recognized and viewed with concern that the webservices paradigm could be used to implement an economic “power grab” strategy through introducing dependencies on centralized resources, similar to how the so-called “Passport” and “.Net My Services” portal was initially intended to be a central aspect of Microsoft's “.Net Initiative” [Ric01a].

This particular sensitivity of Free Software entrepreneurs for risks related to issues of economic power and control is a strong reason to organize an alliance of Free Software businesses as a P2PBA as defined in section 2.3.

4.2 Fair Trade Businesses

Another philanthropic concern which may be a good focus for a P2PBA is to fight for the freedom of all kinds of businesses world-wide to buy and sell on the world market. In this area, the potential entrepreneurs are even less organized than in the case of Free Software businesses, so that their number seems hard if not impossible to estimate, but their number is probably significantly greater because so many people agree on the importance of economic development rural and otherwise disadvantaged areas, especially in the third world. Again the “right to fork” of P2PBAs is important, in this case because otherwise the principle “the one who pays is the one who has the power” would make it difficult for firms in the third world to trust the business alliance to truly act in their best interest.

5.CONCLUSION

We have presented an analysis of the economic needs of business alliances and proposed as suite of webservice protocols which is designed to meet these needs. However this SXDF / QQP / QRPC webservice protocol suite (specified precisely in [Bol04a], [Bol04b], [Bol04c] has performance advantages also in simple situations where the well-publicized combination of XML, HTTP and SOAP is sufficient for meeting the needs.

Therefore it appears desirable to experiment with SXDF, QQP and QRPC for all kinds of webservice protocol needs, and see whether this or a similar approach should perhaps in the long run be advocated as a general standard for webservice interactions. However, SXDF is neither designed nor suitable for persistent storage of user-editable data. If there is interest in using SXDF for such purposes, a companion data format, which may be called ``EXDF'' (Editable Extensible Data Format) will need to be defined in such a manner that conversion between SXDF and EXDF formats can be automated, and EXDF resources can be conveniently edited in any text editor.

6.REFERENCES

[Ban03a] Banan, M.: A vision for libre internet application services, Version 1.2, February 2003. http://www.mailmeanywhere.org/doc.free/neda/libre_vision/philosophy/executiveSummary/one/

[Ber97a] Bernstein, D. J.: Netstrings. February 1997. http://cr.yp.to/proto/netstrings.txt

[Bol04a] Bollow, N.: SXDF - Simple Extensible data Format. Work in progress (published as Internet-Draft in December 2004), http://SXDF.org

[Bol04b] Bollow, N.: QQP – Quick Queues Protocol. Work in progress (published as Internet-Draft December 2004), http://QQP.org

[Bol04c] Bollow, N.: QRPC - Queueable Remote Procedure Calls. Work in progress (published as Internet-Draft December 2004), http://QRPC.org

[Box00a] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S. and D. Winer: Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[Bra00a] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E.: Extensible Markup Language (XML) 1.0 (2nd ed), W3C REC-xml, October 2000, http://www.w3.org/TR/REC-xml

[Coa37a] Coase, R. H.: The Nature of the Firm. Economica **4** (1937): 386-405

[Feh00a] Fehr, E., Gächter, S.: Cooperation and punishment in public goods experiments. American Economic Review **90** (2000): 980-994

[Gov01a] Governor, J.: In praise of the .Net vision. Computing, 11 Jan 2001. http://www.vnunet.com/analysis/1116250

[Int96a] International Organization for Standardization: Information technology - Open Systems Interconnection - Remote Procedure Call ISO/IEC 11578:1996.

[Kle01a] Klensin, J., Ed.: Simple Mail Transfer Protocol, RFC 2821, April 2001, http://www.ietf.org/rfc/rfc2821.txt

[Res01a] Resnick, P., Ed.: Internet message format, RFC 2822. April 2001, http://www.ietf.org/rfc/rfc2822.txt

[Ric01a] Ricciuti, M.: Gates' grand design. CNET News.com, June 2001. http://news.com.com/2009-1082_3-268707.html

[Sta91a] Stallman, R. M.: GNU General Public License, Version 2, June 1991, http://www.gnu.org/copyleft/gpl.html

[Mis99a] Misner, R. I., Devine, V.: The world's best known marketing secret. 2nd ed. Austin, TX 1999 (Bard Press). ISBN 1-885167-37-7.

[Wal99a] Waldo, J.: The Jini architecture for network-centric computing. Communications of the ACM, pages 76--82, July 1999.