SCALABLE, TRUSTWORTHY NETWORK COMPUTING USING UNTRUSTED INTERMEDIARIES

A Position Paper

Micah Beck, Jack Dongarra, Victor Eijkhout, Mike Langston*, Terry Moore and Jim Plank

Department of Computer Science
University of Tennessee

Key Words: scalability, security, collaborative environments, large-scale networks, research/experimental systems

Overview

The extraordinary success of the Internet as a general-purpose, widely sharable, and globally scalable data communication network has set a very high standard for the designers of next generation cyberinfrastructure. Comparable success would be difficult to achieve under any circumstances, but today all visions of major new systems must face intensified requirements for trustworthiness that make this level of success far more challenging to attain. At the heart of that challenge is an apparent conflict between the architectural principles that have made the Internet so amazingly scalable and the main strategies, such as user authentication, that are typically employed to increase the trustworthiness of network information systems and computational grids. The Internet achieves scalability through a modular, bottom-up design in which the critical network layer makes use of untrusted intermediaries and provides a datagram service that gives only the weakest guarantees. By contrast, security and authentication mechanisms often require heavyweight operations (e.g. computationally intensive encryption algorithms) or make strong, shared assumptions that tend to break as the system is scaled up along various important dimensions.

The *Scalable Trustworthy Network Computing (STNC)* project brings together researchers from different areas of Computer Science to explore a novel hypothesis about how this conflict can be addressed:

> ***Using untrusted intermediaries and end-to-end protocols, we can implement a scalable network computing fabric with adequate trustworthiness for a wide spectrum of applications.***

The STNC hypothesis combines a systems element and a combinatorial/numerical element. The systems element is based on the presumption that the Internet model of unbrokered resource sharing can be applied to remote processor resources in order to create a substrate for network computing that can scale the way the Internet scales. Although this approach found initial expression in early grid systems based on remote procedure call mechanisms, such as Dongarra's NetSolve [6, 3], it is now being applied in a more thorough way in the *Logistical Computing and Internetworking (LoCI)* work of Beck and Plank [4, 9]. By applying the Internet design paradigm to the physical resources of storage and computation, the LoCI program aims to create an infrastructure for general distributed systems that is more scalable than anything heretofore achieved. Supposing for the moment, however, that such an approach to scalable computational infrastructure can be achieved, the key question for anyone hoping actually to build and use such a system in today's world is whether or not there is an approach to trustworthiness that can scale along with it.

The idea behind the combinatorial/numerical element of the STNC hypothesis is that there are significant classes of valuable computations that lend themselves to being checked by operations implemented on the end-point or edge systems, using evidence that these systems independently possess. The trustworthiness of this class of computations can be established in a way that is independent of any measures for trustworthiness that have been integrated into a given systems core. Just as the classic Internet design supports reliable and secure data communication by using checksums and encryption implemented in client libraries that operate outside the network core, our proposed research asks how far that same end-to-end design can be generalized to provide independent checks on the results of computational services offered across or within the network. The exploration of this element of the STNC hypothesis will focus on select subclasses of two different types of computations that in different ways fit those criteria:

- $\mathcal{NP}$-complete problems: these problems have the special characteristic of being very difficult to solve exactly, and yet are very easy (i.e. computationally cheap) to check with regard to any purported solution that is offered. In previous work Langston has established ways in which *Fixed-Parameter Tractable (FPT)* problems can serve as

*Communicating author: langston@cs.utk.edu

an important subclass exactly solvable in practice, especially with the aid of the aggregated computing resources available in emerging computational grids [1, 8]. The fact that they can be provided with a definitive check that is inexpensive to apply makes them ideal vehicles for the STNC approach. We will use this software as a proof of concept in our client library for establishing end-to-end trustworthiness, and as the starting point to discover the size and reach of this problem class.

- Linear algebra problems: A wide variety of linear algebra problems, which are ubiquitous in the computational sciences, offer opportunities for a variety of possible strategies that the end-to-end libraries could use to check the putative results of network computations, or even to monitor the correctness of long running computations as they proceed. The quality of check offered ranges along a spectrum from definitive to probable in some lower degree. This area offers a rich space in which to explore the tradeoffs between confidence and economy that applications taking the STNC approach must evaluate.

Under the systems element of STNC, the results of the work in these two areas, along with a variety of well-known system-oriented checks (e.g. voting, spot checking, etc.), will be integrated into a set of end-to-end client libraries that can be used to enhance trustworthy computing in distributed systems of various kinds. Since these end-to-end techniques can be applied independently of any trusted computing measures that have been taken in the network/system core, they can be used generally and as desired to complement other approaches to trustworthy computing. In particular, we believe they can play a very strong role in helping to improve the trustworthiness of a variety of loosely coupled grid systems based on the use of RPC mechanisms [13], such as NetSolve, Ninf [12], and Cactus [2]. While these RPC-based forms of grid computing may have implemented other forms of security, such as Kerberos and Grid Security Infrastructure (GSI) [7], their architectures are well suited to incorporate the STNC libraries, and for some problems those libraries can yield higher levels of confidence than other approaches in the results the system delivers. Our experience in the grid community is that neither Globus nor the GGF is addressing this approach to trustworthiness, nor the scalability benefits that come with it.

For the LoCI program mentioned above the STNC approach is more indispensable. LoCI aims to achieve an extremely scalable infrastructure for network computing by following the Internet design model. An outline of that model can be expressed as follows: Create a basic abstraction of the underlying resource that is as primitive as possible, and then, in order to keep the shared physical substrate passive and free of complexity, require that all stronger services (e.g. reliability, availability, security, fault-tolerance, etc.) be built on top of this primitive service by further aggregation and abstraction *under the control of the end-points or the edge systems*. Thus, in the Internet, reliable data transmission is handled end-to-end by TCP at the transport layer, which, for example, computes checksums to provide a probabilistic error correction. Similarly, if the transmission is to be made secure, higher layer libraries must handle the encryption and decryption end-to-end. The LoCI effort has already successfully applied this paradigm to storage [4, 5, 9] based on a primitive, "best effort" network storage service, and using higher level client libraries to provide for checksum verification of the data retrieved and automatic encryption and decryption of data moving on and off the disk.

But for network computation, simple checksums and encryption simply will not do. The obvious reason is that, unlike data transmission and data storage, network computing services are not expected to hand back exactly what gets put in, so that the character of the result can at best be only partially known in advance. You cannot check the result you receive by computing the identity function. The STNC research proposed under this project aims to find, for the case of network computation, more general replacements for the end-to-end mechanisms that have been used so successfully to make the Internet approach to transmission and storage more trustworthy without sacrificing its wonderful scalability.

Scalability vs. Trustworthiness in Network Computing

There are a number of problems in distributed systems that are "holy grails" for various stakeholders. Highly trustworthy distributed computing is one; highly scalable distributed computing is another. For better or worse, the current state of the world seems to demand that the quest for each must end in a deployable system that somehow incorporates the other. However, beyond the difficulty of achieving either one, there are substantial reasons to think that such a union may be unattainable.

Starting with the problem of trustworthiness, the basis of the usual approach to establishing confidence in the result of a computation performed by code not written by or not executed by the consumer of the result is the use of authentication to establish trust between the user/client and the system/sever. In distributed computing scenarios, the server may be remote and the client and server may not be known to each other. Since there may be no external physical clues to help establish trust, either the client or the server may require the other to authenticate itself. Now it is clear that since authentication at best only establishes confidence in the identity of the other participant in the communication, it has

only an indirect, system-dependent bearing on the trustworthiness of the computation it performs. Reasons that trusted servers may yield untrustworthy results range from programming error or misconfiguration to security breaches of the server or network.

More important to the scalability question, authentication tends to be a heavyweight process: it often uses computationally intensive encryption algorithms, or requires that parties to the transaction agree, in advance, on a trusted third party (e.g. a certificate authority) to mediate the authentication, or requires that the authenticating party consult this third party at each step where there is a possibility of trust revocation. Moreover, the trust conferred by the third party is only as good as the proof used to establish identity to the trusted third party, and achieving high levels of trust can require physical inspection. Finally, the veracity of the use of certificates is a function of the competence and honesty of unknown system administrators, who cannot easily be made party to the authentication process. Because of this tendency toward heavyweight complexity, there is a high cost associated with the (indirect) confidence that trusted computing infrastructures, such as computing and data grids, confer on the results they deliver; they do not scale well, at least not by the standards that the Internet has encouraged us to expect.

As is evident in the formulation of the STNC hypothesis stated above, our research approaches the problem from the other direction. To achieve a synthesis of scalability and trustworthiness, we follow the lead of the networking community in applying the End-to-End Principles to the problem. In the form in which their authors have most recently expressed them [10], these principles state that in order to achieve scalability:

1. Intermediate nodes should be general, and should not implement specialized functions to support subclasses of applications.

2. Intermediate nodes should not implement functions that can be implemented at the end-points.

Trying to achieve high confidence in results by injecting trust into the core of the system from which they issue is a typical example of the perils of ignoring the second principle: if confidence in the result is established in any agent other than the client itself, then transmitting the result, along with the associated confidence, requires trust in remote processes; such a situation may possibly succeed in controlled environments, but not in a globally scalable network.

But the price for Internet levels of scalability achieved through these principles is also quite high. They require that the reliability and security mechanisms employed be external, or system independent, so that they can be implemented entirely at the endpoints. Thus reliability checks through the use of checksums and privacy measures in the form of end-to-end encryption are used to supply a basic level of trustworthiness for data communication. Stronger forms are available only if they can be done end-to-end. The question that obviously needs an answer is whether and to what degree such external, system-independent mechanisms are available for and adequate to the challenges of network computing, especially when the aim is to scale to the wide area.

To motivate our answer to this question, let us focus for a moment on just the problem of establishing confidence in the reliability of remote computation, leaving the discussion of privacy measures to a subsequent section. Clearly, if the end-point is to establish confidence in the result in a system-independent way, then it must have its own evidence of the result's correctness. Approaches to this problem that use *redundancy*, either in the form of voting or spot checks [11], are well known and widely used (e.g. seti@home, folding@home). Likewise *internal consistency checks* can be used to test for errors in computation and data transmission, as checksums are used by the Internet as a form of probabilistic internal consistency check. But the highest level of confidence is established by *external consistency checks*, where the results of remote computations are shown to be consistent with external data that the client holds, such as the input to that computation.

Although the STNC project embraces weaker end-to-end measures, we want to explore in depth the notion of using scalable, external consistency checks for trustworthy network computing. So consider, for example, a generalized scenario in which an application program has some task that must be performed, but that it cannot or will not perform it by itself. Instead, the program delegates the task to some system component or network node. These nodes may be untrusted. It cannot be assumed that they will perform the task correctly, or even that they will keep the task confidential while it is being performed. There are several special cases to this scenario.

- It may be that the task requires data to be delivered to a particular destination. In this case, data transmission paths and IP routers are considered untrusted. This assumption is central to achieving scalability in the network. Reliability is achieved by using checksums: the sender augments the data with a checksum to create internal redundancy and the receiver checks for the consistency of that redundant information. Any error in data transmission will, with high probability, cause inconsistency between the data and checksum, and the data will be rejected. Malicious errors and theft of the data are avoided using encryption: if the data with checksums is encrypted by the sender, then

malicious errors that would evade a checksum that is visible in the clear will cause inconsistencies in the data when decrypted.

- It may be that the task causes data to be stored on a disk. In the case of storage, disks and file systems are sometimes considered untrusted because of the possible degradation of data on the disk, or the possibility of operating system security mechanisms being compromised. The main solutions in this case are identical to the data transmission case: checksums and encryption.

- Or it may be that the task is a call out, so that data must be transformed through computation, producing some form of output. In the case of computation, the equivalent of the receiver verifying a checksum in the network case is for the caller to verify the return value by applying some check. In some cases the return value is augmented with redundant information to make the checking more efficient. Some checks are probabilistic (like checksums in the network case) and some are absolute, ensuring that the computation was correct.

An example of a computation that can be checked is matrix inversion. If a call-out is made to invert a matrix A, then the result B can be multiplied by A and the product AB compared to the identity matrix. This check is absolute, because the correctness of the result is established in a non-probabilistic fashion. A probabilistic version of this case would be for randomly selected elements of the product AB to be calculated. If all selected elements were correct, this would only indicate the probability that all elements were correct. Another example of an absolute check is when the computation is to solve an instance of an $\mathcal{NP}$-complete problem. In this case, membership in $\mathcal{NP}$ ensures that the caller can check, in deterministic polynomial time, any solution tendered by the receiver.

Thus, checking can be used to ensure that the result of a call-out is correct; keeping the computation secure is a harder problem. Clearly, a malicious call-out server can steal the argument and the results, and then reveal to others information that the caller wants to keep private. The use of encryption in this case is difficult, because the decrypted form of the data must never be revealed to the call-out server; the computation must proceed on encrypted data. Finding methods of encryption that can be composed with computation to yield the desired result is a formidable task. In some cases, it may be acceptable to settle for a very light encryption that obscures the computation from casual observation, but not from an attacker determined to discover it.

What these examples make clear is that, unlike voting and spot-checks, the strong yet efficient consistency-based approaches must take advantage of knowledge about the function being computed. Thus, if we are to follow the first of our end-to-end principles, our system will either require that the user supply both the code for the function and the consistency check, or else pre-installed code must be used to implement a set of functions that is sufficiently general. While both cases are of interest, the latter case provides an opportunity to apply formal methods for creating a general computing infrastructure that is conveniently usable to a wide spectrum of clients with no need for them to apply formal reasoning, much as is the case with Public Key Infrastructure currently in use for the purposes of authentication.

All of these end-to-end approaches — voting, spot checking, internal and external consistency checks — can be applied in a probabilistic manner, with generally increasing costs for approaches that build higher confidence. For some problems and methods, such as establishing confidence in arbitrary data passed across the network, only probabilistic approaches are possible. Particularly with probabilistic approaches, a combination may yield the best resilience to a variety of possible sources of error. For this reason, our experimentation will integrate all feasible approaches. However, the core of our research program is based around consistency-based approaches, and in particular on strong algorithms based on external consistency. This is where we will concentrate our efforts under STNC.

Now *if* there are valuable kinds of computation that lend themselves to external consistency checks (a condition which our research should confirm), it is clear that the advantages of having such functionality are extremely strong:

1. External consistency checks, using provably valid methods, is the only mechanism that can be used to establish confidence in the result of a remote computation that is equal to the confidence that the client has in its own local computations. It is arguably the best kind of corroboration that it is humanly possible to obtain.

2. Where such checks are possible, they provide an independent way of validating the trustworthiness of authenticated systems. Therefore they do not compete with conventional systems based on authentication, but rather provide a valuable complement to them.

3. Finally, since such end-to-end checks add no complexity to and place no burden on the shared system infrastructure, they do not inhibit or degrade the scalability of a system that uses them. As noted above, in our LoCI research we are exploring an approach to highly scalable network computing that could be deployed for a set of problems whose range and significance is as broad as the STNC research proposed here can establish.

The obvious weakness of current formulations of corroborated and encrypted computing methods is that they are not general, applying only to certain classes of computation performed remotely. In the research described in the sequel, we will show that there are extremely valuable families of computation that fit this approach well. While we believe our work will show that the range of usable computations can be greatly broadened, there is admittedly no guarantee that this strategy can ever be completely generalized.

By starting with mechanisms that we know can provide truly reliable and secure computing without obstructing scalability we can be certain that, to the limits by which our techniques are generalizable, they provide the most important characteristic: a knowable degree of trustworthiness. By working up from the specific cases in order to expand the class of problems that fit our framework, we will be working to increase the usefulness of system mechanisms that always meet the correctness criteria, and which can be integrated into a wide variety of system types. While our work proceeds, we will always be able to make provable assertions about methods we have developed. Any application that can make use of them (e.g. through the software libraries we create) will achieve unequaled level of trustworthiness, and any systems that employ them will not have their scalability thereby degraded.

References

[1] F. N. Abu-Khzam, M. A. Langston, and P. Shanbhag. Vertex cover: A case study on the practical feasibility of exact algorithms for fixed-parameter tractable problems. Technical Report UT-CS-02-494, Department of Computer Science, University of Tennessee, 2002.

[2] Gabrielle Allen, Werner Benger, Tom Goodale, Hans-Christian Hege, Gerd Lanfermann, Andre Merzky, Thomas Radke, Edward Seidel, and John Shalf. The cactus code: A problem solving environment for the grid. In *HPDC*, pages 253+, 2000.

[3] D. C. Arnold, S. S. Vahdiyar, and J. Dongarra. On the convergence of computational and data grids. *Parallel Processing Letters*, 11(2):187–202, 2001.

[4] M. Beck, T. Moore, and J. S. Plank. An end-to-end approach to globally scalable network storage. In *ACM SIGCOMM '02*, Pittsburgh, August 2002.

[5] M. Beck, T. Moore, J. S. Plank, and M. Swany. Logistical networking: Sharing more than the wires. In *Active Middleware Services*. Kluwer Academic Publishers, 2000.

[6] H. Casanova and J. Dongarra. Applying netsolve's network enabled server. *IEEE Computational Science & Engineering*, 5(3):57–66, 1998.

[7] Ian T. Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[8] J. M. Lehrter, F. N. Abu-Khzam, D. W. Bouldin, M. A. Langston, and G. D. Peterson. On special-purpose hardware clusters for high-performance computational grids. In *Proceedings, International Conference on Parallel and Distributed Computing and Systems*, pages 1–5, 2002.

[9] J. S. Plank, A. Bassi, M. Beck, T. Moore, M. Swany, and R. Wolski. Managing data storage in the network. *IEEE Internet Computing*, 5(5):50–58, 2001.

[10] D. P. Reed, J. H. Saltzer, and D. D. Clark. Comment on active networking and end-to-end arguments. *IEEE Network*, 12(3):69–71, 1998.

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

[12] Mitsuhisa Sato, Hidemoto Nakada, Satoshi Sekiguchi, Satoshi Matsuoka, Umpei Nagashima, and Hiromitsu Takagi. Ninf: A network based information library for global world-wide computing infrastructure. In *HPCN Europe*, pages 491–502, 1997.

[13] K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee, , and H. Casanova. Overview of GridRPC: A remote proceedure call API for grid computing. In *Grid Computing – GRID 2002, LNCS*, volume 2536, pages 274–278. Springer Verlag, 2002.