1. Uncertainty and Predictability: Can they be reconciled? *

Paulo Veríssimo

Univ. of Lisboa, Faculty of Sciences
Lisboa - Portugal
Email: pjv@di.fc.ul.pt
http://www.navigators.di.fc.ul.pt

1.1 Introduction

We are faced today with the confluence of antagonistic aims, when designing and deploying distributed systems. On one hand, our applications have to achieve timeliness goals, dictated both by QoS expectations with regard to on-line services (e.g. time-bounded transactions), and by technical issues of real-time nature involved in the deployment of certain services (e.g., multimedia rendering). On the other hand, the open and large-scale environments where applications and users execute and evolve exhibit uncertain timeliness or synchrony. Likewise, services, despite their sometimes critical nature (not only money-critical, but also privacy- or even safety-critical), are more often deployed on-line or through open networks. It is required that they be resilient to intrusions, despite the elusiveness of attacks they are subject to, and the pervasiveness and subtelty of vulnerabilities in the relevant systems. In other words, the environment in which these services have to operate exhibits uncertain behavior: we cannot predict all possible present and future attacks; we cannot diagnose all vulnerabilities.

In the previous paragraph, we essentially talked about *uncertainty*, the grand challenge faced by distributed system researchers and designers. When talking about uncertainty, 'impossibility' and 'probability' are words that come to mind. Literature has relevant examples on being pessimistic and accepting uncertainty, showing impossibility results[1.1], or producing solutions that are uncertain, albeit quantifiably uncertain [1.2, 1.3]. Other works have methodically studied what can be done when the system is incrementally less uncertain[1.4]. Alternatively, other approaches are more optimistic, assuming that the system has periods of determinism, alternating with uncertainty, and try to identify and successfully explore those (sometimes scarce) periods, to perform useful tasks[1.5, 1.6].

Nevertheless, a designer does not make strong assumptions about synchrony, or security, or structure, just for the sake of it. They are made because

* Work partially supported by the EC, through proj. IST-1999-11583 (MAFTIA), IST-FET-2000-26031 (CORTEX), and FCT, through the Large-Scale Informatic Systems Laboratory (LaSIGE) and proj. POSI/1999/CHS/33996 (DEFEATS).

they provide guarantees (read: combinations of timeliness and reliability) not enjoyed by other alternatives, in essence, a degree of *predictability* about system attributes. Recently, we observe works which make increasingly strong assumptions about the environment, to get correspondingly higher guarantees. For example, arguing about the advantage of having perfect failure detectors[1.7]. Such failure detectors, however, cannot be implemented on environments with uncertain synchrony, which have been the workhorse of all past work on failure detectors. This status quo might be extended to security and survivability: giving hard guarantees on security of services often requires strong properties of the underlying environments.

On the more practical side, designers of protocols and systems like Squid, Akamai, Inktomi, AOL, etc., confronted for example with the uncertainty of the Internet, have been providing ad hoc (but normally extremely effective) mechanisms, such as warm cache hierarchies or prioritary execution of special tasks, for performance, or dedicated channels (e.g., physical or overlay networks), both for performance and security. However, short of a systemic approach to the problem, guarantees are essentially statistical. This would be enough for those applications, because service provision has largely been based on average performance and loose contractual guarantees, but has not solved the predictability problem, for example: the latency of individual transactions, or multimedia frames, or the intrusion tolerance of a TTP server.

In this paper, we discuss a novel design philosophy for distributed systems with uncertain or unknown attributes, such as synchrony, or failure modes. This philosophy is based on the existence of architectural constructs with privileged properties which endow systems with the capability of evading the uncertainty of the environment for certain crucial steps of their operation where predictability is required. It may open new research avenues allowing to reconcile uncertainty with predictability.

1.2 The Wormhole metaphore

So, what system model and design principles will allow us to meet the grand challenge posed by uncertainty? We propose a few guiding principles: assume that uncertainty is not ubiquitous and is not everlasting— the system has parts more predictable than others and tends to stabilize; be proactive in achieving predictability— make it happen at the right time, right place.

In what follows, we wish to share with the reader one possible research track that follows the above-mentioned guidelines. We introduce it with the help of a metaphor. In the universe, speed of light is the fastest that can be attained, which would make it impractical to travel to or communicate with remote parts of the universe. However, a theory argues that one could take shortcuts, through, say, another dimension, and re-emerge safely at the desired point, apparently much faster than what is allowed by the speed of light. Those shortcuts received the inspiring name of *wormholes*.

Let us move from metaphor to reality. Assume that we can construct a *distributed system with wormholes*. There would be the 'payload system' where applications execute, i.e. the "normal" system with several hosts interconnected by the 'payload network', the usual Internet/Intranet. Then, there would be a small alternative subsystem whose behavior would be predictable: the 'wormhole subsystem'. This subsystem would be accessed from the payload through 'wormhole gateways', devices local to hosts. In practical terms, the wormhole is an artifact to be used only when needed, and its services supposedly implement functionality hard to achieve on the payload system, which in turn should run most of the computing and communications activity.

So, in conclusion, the key characteristic of the architecture of a system with wormholes consists in assuming that, no matter how uncertain the system and its behavior may be, there will be a subsystem which has 'good', well-defined properties. This subsystem is small and simple, so that its construction with trustworthy behavior is feasible. Note that whilst the most fascinating and powerful incarnation of a wormhole would be distributed, we can envisage simpler versions, with local (non-networked) wormholes, which still provide very useful support (e.g., local security or timeliness functions).

1.3 Is it possible to travel through Wormholes?

This metaphoric question translates into two practical ones:

Is it feasible to construct systems such as postulated above?

Are systems with wormholes of any real use?

As to the construction, observe Figure 1.1, where we suggest two possible implementations, one for small-area settings, another for wide-area ones. The first, in Figure 1.1(a), shows a mission critical web server replicated inside a facility, such as a Command, Control and Communications Center. The local wormholes can be implemented by some sort of appliance board with a private network adapter. The wormhole interconnection inside a facility as in the example can be implemented by a private LAN interconnecting the wormhole adapters. Figure 1.1(b) shows an authentication service distributed over a wide area. The wormhole interconnection in this case has to be highly secure, deterministic and work in wide-area. A feasible example of the above is depicted in the figure: wide-area Virtual Private Networks (VPN), constructed over ISDN or 3G-UMTS.

Figure 1.2 shows less trivial examples of the utility of wormholes. Figure 1.2(a) suggests the use of wormholes to enhance the control of overlay networks (ON). In fact, wormholes should not be confused with ONs, but the latter could be built on the wormhole concept, to strengthen its predictability, as the figure suggests. The payload channel would be implemented with the normal ON network support, whilst a control channel with differentiated

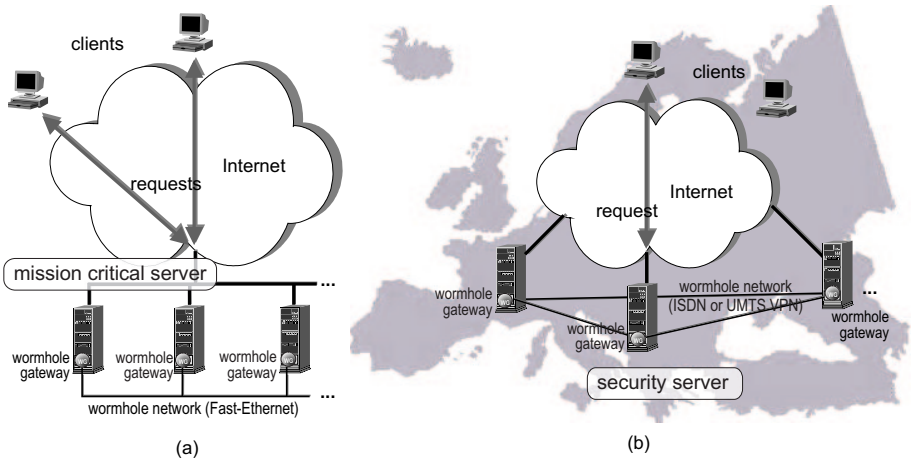

Fig. 1.1. Examples of real systems with wormholes: (a) replicated mission critical web server; (b) distributed security server

properties would ensure predictable (timely and secure) management and reconfiguration of the ON. Figure 1.2(b) depicts a situation where a team of mobile units is moving and occasionally hooking to wired base stations. Imagine for example a platoon of cars or a team of robots. It is important that cars/robots keep timely and coherent synchronization, despite any glitches in their wireless payload networking support. The wormhole would help achieve that objective.

As to the usefulness of wormholes, consider one instance of the global problem we stated: *performing timely actions in the presence of uncertain timeliness*. In one of our experiments, we have prototyped a specific kind of wormhole subsystem for achieving predictable behavior in systems of uncertain synchrony, anywhere in the spectrum from asynchronous to synchronous[1.8]. We called it the *Timely Computing Base*. In [1.9] we present a formal embodiment of the model. The Timely Computing Base can for example be used to build perfect failure detectors and thus support asynchronous algorithms running on the payload system and relying on the former detectors[1.7].

In the malicious failure domain, the potential for intrusion tolerance can be drastically augmented by using wormholes, for two reasons: they implement some degree of distributed trust for low-level operations, acting as a *distributed security kernel*; they give more room for the uncertainty of malicious behavior in the payload system. In a second experiment, we showed a way of *performing trusted actions in the presence of uncertain attacks and vulnerabilities*. We devised a set of new functions resilient to malicious faults for this new wormhole, calling it Trusted Timely Computing Base[1.10].

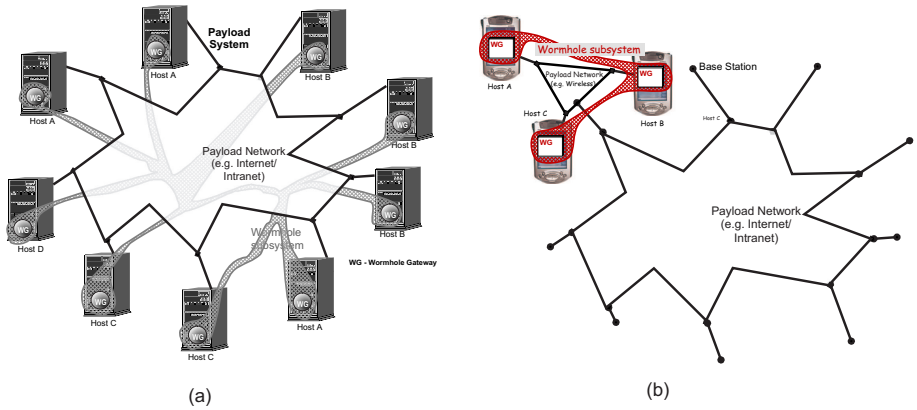

Fig. 1.2. Examples of real systems with wormholes: (a) Enhancing overlay network control; (b) Supporting mobile teams

1.4 Conclusions and Future Work

We have given a unifying perspective to a recent research effort around a novel approach to distributed systems design. This work was triggered by the intutition of the need to handle uncertainty but still be able to provide predictable behavior. The approach is well-founded, since by introducing the necessary architectural devices— *wormholes*— the desired behavior is enforced, rather than assumed. The example implementations shown address scale from a limited perspective: wormholes can be deployed in large-scale in terms of geographical scope— relatively few but very far apart— and communities of few can serve collections of very many— client-server. However, as a concept, wormholes need not be limited by scale, if the following challenge is solved: how to predictability communicate from many to many, using scarce resources.

Authors are looking for better algorithms, more efficient, faster or even timed. Most of these works require constructs that prefigure the concept of wormhole[1.7, 1.11]. Moreover, a recent paper has shown that "there is no free lunch"[1.12]: if we wish to do really useful things, in the presence of an unbounded number of failures (or uncertain, for the matter), we have to make correspondingly strong assumptions about our environment. In the cited paper, the authors argue about the need for perfect failure detectors (and no weaker). Such failure detectors could be easily implemented on environments as postulated in this paper. More recently[1.13], authors are proposing to study efficient schemes for using wormholes— since they are a scarce resource.

It is our intention that the ideas presented here are used as a foundation to build systems meeting the present and future challenges concerning

uncertainty. As a matter of fact, most of our recent work was focused on consolidating this foundation, and producing the prototypes we have shared with the community (**http://www.navigators.di.fc.ul.pt/software/tcb**). Challenging payload protocols can be built using wormholes, and we are just starting to discover that[1.14]. We plan on further exploiting the power of wormholes in areas as different as: intrusion-tolerant consensus and interactive consistency; timed agreement and ordering primitives; event-based communication for cooperative and embedded mobile systems.

References

1.1 Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM **32** (1985) 374–382
1.2 Rabin, M.O.: Randomized Byzantine Generals. In: Procs. of the 24th Annual IEEE Symposium on Foundations of Computer Science. (1983) 403–409
1.3 Cristian, F.: Probabilistic Clock Synchronization. Distributed Computing, Springer Verlag **1989** (1989)
1.4 Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM **35** (1988) 288–323
1.5 Chandra, T., Toueg, S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM **43** (1996) 225–267
1.6 Christian, F., Fetzer, C.: The timed asynchronous system model. In: Proceedings of the 28th IEEE International Symposium on Fault-Tolerant Computing. (1998) 140–149
1.7 Helary, J., Hurfin, M., Mostefaoui, A., Raynal, M., F., T.: Computing global functions on asynchronous distributed systems with perfect failure detectors. IEEE Transactions on Parallel and Distributed Systems **11** (2000)
1.8 Veríssimo, P., Casimiro, A., Fetzer, C.: The timely computing base: Timely actions in the presence of uncertain timeliness. In: Procs. of the Int'l Conference on Dependable Systems and Networks, New York City, USA (2000) 533–542
1.9 Veríssimo, P., Casimiro, A.: The timely computing base model and architecture. IEEE Transactions on Computers, Special Issue on Asynchronous Real-Time Distributed Systems (2002)
1.10 Correia, M., Veríssimo, P., Neves, N.F.: The design of a COTS real-time distributed security kernel. In: Proc. of the Fourth European Dependable Computing Conference, Toulouse, France (2002)
1.11 M. Aguilera, G.L.L., Toueg, S.: On the impact of fast failure detectors on real-time fault-tolerant systems. In: Proc. of DISC 2002. (2002)
1.12 Delporte-Gallet, C., Fauconnier, H., Guerraoui, R.: A realistic look at failure detectors. In: Proceedings of the International Conference on Dependable Systems and Networks, Washington, USA (2002) 213–222
1.13 R. Friedman, A. Moustefaoui, S.R., Raynal, M.: Error correcting codes: A future direction to solve distributed agreement problems? In: International Workshop on Future Directions of Distributed Computing, FuDiCo. (2002)
1.14 Correia, M., Lung, L.C., Neves, N.F., Veríssimo, P.: Efficient byzantine-resilient reliable multicast on a hybrid failure model. In: Proc. of the 21st Symposium on Reliable Distributed Systems, Suita, Japan (2002)