# A Wormhole-based Intrusion-Tolerant Group Communication System – WIT-GCS

(Extended abstract) *

Miguel Correia[1] Nuno Ferreira Neves[1] Lau Cheuk Lung[2] Paulo Veríssimo[1]

[1] *Faculdade de Ciências da Universidade de Lisboa*

Bloco C5, Piso 1, Campo Grande, 1749-016 Lisboa - Portugal

[2] *Pontifícia Universidade Católica do Paraná*

Rua Imaculada Conceição, 1155, Prado Velho - Brasil - CEP: 80215-901

{mpc,nuno,pjv}@di.fc.ul.pt lau@ppgia.pucpr.br

## 1 Introduction

Group communication is a well-known paradigm for the construction of distributed applications. This abstract is about the design of a Wormhole-based Intrusion-Tolerant Group Communication System – WIT-GCS. The system is *intrusion-tolerant* in the sense that it tolerates arbitrary faults, including both accidental and malicious faults such as attacks and intrusions [12]. The system is expected to continue to provide correct results despite intrusions on a number of processors and attacks in the network, e.g., delay, modification, or replay of messages. The system is composed by a *membership service* and a *view-synchronous atomic multicast primitive*. WIT-GCS is as instantiation of the MAFTIA project middleware architecture [1].

The intrusion-tolerant GCSs in the literature assume a homogeneous system, i.e., that all components can be attacked and fail, and all processing delays and message delivery delays are equally unknown (except for occasional synchrony assumptions) [9, 6, 7, 8]. This abstract is based on a different approach. We consider that most of the system has these same characteristics: insecure and with uncertain timeliness (although we also need a synchrony assumption). However, we assume that this system is extended with a privileged distributed component, which is secure and real-time (capable of executing timely operations). This new kind of privileged components are called *wormholes* [11].

The abstract explores a specific wormhole called Trusted Timely Computing Base (TTCB), which has innovative characteristics: it is distributed with its own secure communication channel; it is secure and can only fail by crashing; it is synchronous, capable of timely behavior; and it provides a small set of services that can be used to implement intrusion-tolerant protocols. A system with a TTCB is depicted in Figure 1. The TTCB is represented in white. The design of this wormhole was presented in [5], and a COTS-based implementation is currently available for free non-commercial use[1].

WIT-GCS runs to most extent in the 'outside' system, called payload system: processors and payload network in the figure. It uses the TTCB only to execute a few crucial steps of the protocols. Why not run the whole system inside the wormhole, since it is secure anyway? The basic argument about the TTCB is that it is possible to implement a secure and real-time distributed component if and only if it is 'simple' and 'small'. Therefore, only a small number of services is provided and the resources available to run these services are limited.

*This work was partially supported by the EC, through project IST-1999-11583 (MAFTIA), and by the FCT, through the Large-Scale Informatic Systems Laboratory (LASIGE) and projects POSI/1999/CHS/33996 (DEFEATS) and POSI/CHS/39815/2001 (COPE). This extended abstract resumes the work reported in [4].

[1] http://www.navigators.di.fc.ul.pt/software/tcb/

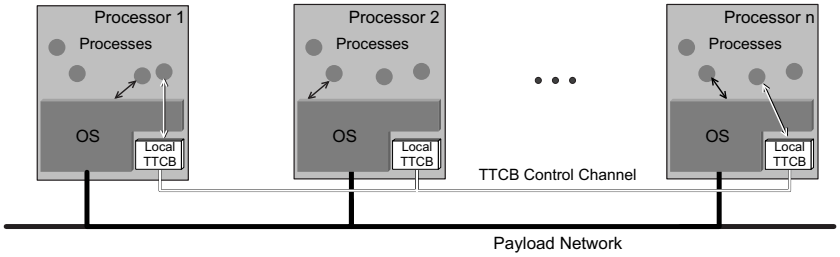


Figure 1: Architecture of a system with a TTCB.

## 2 Membership Service

A membership service handles basically three operations: the addition of members to a group, the removal of failed members, and the removal of members by their own initiative. These operations will be called respectively join, remove and leave. The failure of a processor is detected in every processor by a failure detector module. The GCS assumes that $f \leq \lfloor \frac{|V^n|-1}{3} \rfloor$ processors can fail in a given membership $V^n$, i.e., less than one third of the processors.

The WIT-GCS membership service generates *views*, i.e., numbered events containing the group membership. A new view is installed whenever the membership is changed due to a member join, leave or removal. A processor $P_j$ sees a view as an array $V_j^n$ containing one entry per each member processor. The index $n$ reflects the $n^{th}$ view of the group. The service guarantees that each correct processor has the same view at every instant of logical time, i.e., after the installation of the same totally ordered views in every processor.

The membership service is implemented by the Membership and Message Delivery protocol (MMD). MMD is a finite state machine that evolves at each processor in two states: Normal and Agreement. When a processor joins a group it enters the Normal state. This is the state where the system is supposed to be most of the time, and where processors may communicate normally. Then, when another processor wants to join or leave, or when a processor is suspected to have failed, certain *events* are generated and the protocol changes to the Agreement state. In this state, the processors of the current view try to agree on the next view, by running the View Change and Message Delivery Agreement protocol (VCMDA). When VCMDA terminates, the new view is installed and the state changes back to Normal.

We do not present the pseudo-code of the protocols for lack of space (see [4]). Figure 2 presents an example execution instead. Initially, the group has four processors, $P_1$ to $P_4$. $P_4$ is malicious and performs malicious actions that are detected by the failure detectors of processors $P_1$ and $P_2$. When this happens, $P_1$ and $P_2$ multicast a (INFO, Remv($P_4$)) message saying that $P_4$ should be removed from the group. Even if $P_3$ does not detect the failure of $P_4$, when it gets $f + 1 = 2$ messages stating that $P_4$ should be removed, it knows that at least one correct processor detected the failure, since at most $f = 1$ processors can fail and "lie". Therefore, when $P_3$ receives the second (INFO, Remv($P_4$)) message it also multicasts the same information.

When a processor receives $2f + 1 = 3$ messages saying $P_4$ failed it can be sure that all correct processors will also receive 3 or more messages [4]. Therefore it can move to the Agreement state with the confidence that all correct processors will eventually do the same. It can put Remv($P_4$) in a bag called bag-decisions, where it saves all the changes that have to be applied to the current view, also knowing that all correct processors will do the same. The bag is used to store all events that have to be agreed upon by the protocol.

In the Agreement state the processors execute the VCMDA protocol. The objective is to make all correct processors decide the same changes to the view. The protocol relies on the TTCB *Trusted Block Agreement service* (TBA) to agree on the view changes. TBA is the main service used to support the execution of intrusion-tolerant protocols. It delivers a value obtained from the agreement (in a broad sense) of the values proposed by a set of software entities. The values are blocks with a limited size, and for this reason the service is not intended to do all agreement operations in a system, but only to perform some steps of the protocols.

We have no space to delve into the details of the service. The important is that VCMDA uses the TBA service to agree on a digest of bag-decisions. In the example, $P_1$ to $P_3$ propose identical digests – they have the same Remv($P_4$) event in the bag – and TBA returns that digest, since it decides the most

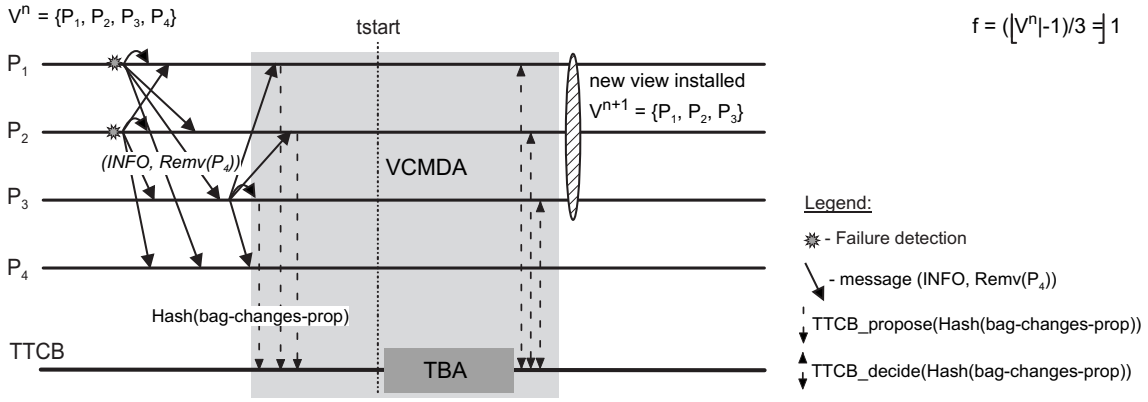


Figure 2: Membership service example execution.

proposed value. Next, the new view is installed and $P_4$ is removed.

# 3 View-Synchronous Atomic Multicast

The View-Synchronous Atomic Multicast protocol (VSAM) provides a view-synchronous semantics, i.e., it guarantees that all correct group members deliver the same messages in the same view [2]. Group communication usually involves a set of multicast primitives with different order properties. VSAM orders messages in total order, i.e., all correct processors deliver the messages in the same order.

VSAM relies on an reliable multicast protocol to guarantee that all processors eventually receive the same messages [4, 3]. When a processor receives a number of messages, MMD starts VCMDA that decides which messages to deliver. The delivery order is given by a timestamp taken from the TTCB, which has synchronized clocks.

# 4 Conclusion

This abstract explores a novel system model. The system is basically asynchronous and vulnerable to arbitrary faults, including attacks and intrusions, but it includes a distributed trusted and real-time subsystem called TTCB. This subsystem is an example of a wormhole, a privileged component that provides limited but useful services for applications and protocols otherwise executed in the normal weak environment.

The WIT-GCS system is composed by a membership service and a view-synchronous atomic multicast protocol, VSAM. By relying on the TTCB wormhole these two components manage to have interesting features, when compared with similar systems in the literature.

Firstly, the system seems to perform considerably better. The system model and experimental settings used in the evaluations of similar systems in the literature were different from ours so comparisons have to be made with caution. Currently, we are aware of only three other implementations of membership services for systems that might experience Byzantine faults, which are Rampart [10], ITUA [8] and SecureRing [6]. We used machines faster than Rampart and SecureRing, but slower than ITUA, and our numbers are about 10 to 20 times better. There are also some numbers available for the performance of intrusion-tolerant view-synchronous atomic multicast protocols in Rampart and ITUA. The latency and throughput of WIT-GCS seems to be approximately 10 times better that those systems. Additionally, VSAM also does not degrade its performance with the number of processors involved, although only a limited number of machines was available.

The second benefit of WIT-GCS is that it makes decisions in a distributed way instead of relying on a leader. This is a considerable advantage for two reasons. The first is that detecting a malicious leader is an operation that can cost some time, therefore a failed leader delays the protocol. The second is that an attacker can try to delay the service by postponing the communication and creating false failure suspicions of successive leaders.

Finally, to ensure the termination of the VCMDA protocol, it is necessary only to make a weak synchrony assumption about the execution of the processors. We say the assumption is 'weak' because it has only to eventually occur and it is not about the communication in the network, but only about the processors. The protocol, however, was built in such a way that even if this assumption is never verified, it never violates the safety properties.

The benefits of WIT-GCS can give the reader an intuition about the practical interest of using wormholes in the context of intrusion tolerance. Current and future work involve the design of new wormhole-based systems and the search for new benefits this architecture may provide.

# References


[1] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J. C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds, R. Stroud, P. Veríssimo, M. Waidner, and A. Wespi. *Conceptual Model and Architecture of MAFTIA. Project MAFTIA deliverable D21.* January 2002. http://www.research.ec.org/maftia/deliverables/D21.pdf.

[2] K. Birman and T. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1):46–76, February 1987.

[3] M. Correia, L. C. Lung, N. F. Neves, and P. Veríssimo. Efficient Byzantine-resilient reliable multicast on a hybrid failure model. In *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems*, pages 2–11, October 2002.

[4] M. Correia, N. F. Neves, L. C. Lung, and P. Veríssimo. WIT-GCS – a wormhole-based intrusion-tolerant group communication system. Submitted for publication, 2003.

[5] M. Correia, P. Veríssimo, and N. F. Neves. The design of a COTS real-time distributed security kernel. In *Proceedings of the Fourth European Dependable Computing Conference*, pages 234–252, October 2002.

[6] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. The SecureRing group communication system. *ACM Transactions on Information and System Security*, 4(4):371–406, November 2001.

[7] L. E. Moser, P. M. Melliar-Smith, and N. Narasimhan. The SecureGroup communication system. In *Proceedings of the IEEE Information Survivability Conference*, pages 507–516, January 2000.

[8] H. Ramasamy, P. Pandey, J. Lyons, M. Cukier, and W. H. Sanders. Quantifying the cost of providing intrusion tolerance in group communication systems. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 229–238, June 2002.

[9] M. Reiter. Secure agreement protocols: Reliable and atomic group multicast in Rampart. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 68–80, November 1994.

[10] M. K. Reiter. A secure group membership protocol. *IEEE Transactions on Software Engineering*, 22(1):31–42, January 1996.

[11] P. Veríssimo. Uncertainty and predictability: Can they be reconciled? In *Future Directions in Distributed Computing*, volume 2584 of *Lecture Notes in Computer Science*, pages 108–113. Springer-Verlag, 2003.

[12] P. E. Veríssimo, N. F. Neves, and M. P. Correia. Intrusion-tolerant architectures: Concepts and design. In R. Lemos, C. Gacek, and A. Romanovsky, editors, *Architecting Dependable Systems*, volume 2677 of *Lecture Notes in Computer Science*, pages 3–36. Springer-Verlag, 2003.