MilCAN and Ethernet

Periklis Charchalakis, George Valsamakis, Bob Connor, Elias Stipidis

This paper aims to give a general description of a Bridge designed to transparently interconnect a cloud of MilCAN [1] segments using a high-speed backbone such as Ethernet. MilCAN is a protocol developed under the auspices of the International High Speed Data Bus User's group in order to cover the requirements for military Vetronics systems, providing deterministic and redundant communication between MilCAN nodes. With the constant increase of embedded devices within a vehicle the need for interconnection clashes with the speed limits of available backbone protocols. The VSI-Bridge is a software-based Bridge with routing capabilities (BRouter) developed to connect a MilCAN segment to high-speed complex protocols used as backbones. The target was to offload non-critical traffic from the embedded network to the backbone, thus reducing the load.

Introduction

Military Vetronics systems are predominantly distributed real-time systems where the communication network has to provide a reliable and deterministic transport mechanism. The whole system has to be deterministic, redundancy capable, and with error detection and isolation.

The MilCAN protocol, developed within QinetiQ (formerly DERA), provides a suitable solution for Vetronic systems. Using the CANbus network, MilCAN adds a thin Quality of Service layer below the application. By implementing message priorities, message 'time to live' and time synchronization between the nodes, the traffic on the CAN bus is controllable and the network deterministic, without the need for a traffic management scheme within the application.

As MilCAN is based on CANbus it is limited by its theoretical maximum 1Mbit bandwidth. When multiple segments need to be interconnected with a backbone this becomes a problem, as the determinism of the network will be affected by the high traffic. The use of MilCAN-to-MilCAN bridges that filter unwanted traffic could provide a solution, but again the backbone bandwidth usage will have to be kept well below the 1Mbit limit, thereby limiting the number of connected segments (depending on the application). This would also make it difficult to monitor the ability of real-time traffic of a segment the backbone.

Using a high-speed network as a backbone would solve the bandwidth limitation and allow future addition of more segments, without the need to reconfigure the current network. Using Ethernet for backbone provides a viable solution where the high bandwidth (up to 10Gbps) can support a very high number of MilCAN segments, and also compensate in some degree to the lack of determinism.

The VSI Bridge, developed in collaboration between the Vehicle Systems Integration (VSI) group at QinetiQ and the Communication Research Group (CRG) of Sussex University, is a Proof of Concept implementation of using Ethernet for transparently bridging multiple MilCAN segments. It was designed to maintain the priority scheme of MilCAN both internally (Bridge components) and externally (network interfaces). It also implements a Standard Interface Layer, where one or more CAN, Ethernet, or any other network, could be added at any time and interoperate transparently with each other. In addition, routing capabilities and a remote configuration utility provide easy network management and monitoring to the user.

MilCAN Protocol

The MilCAN operation is based on bus scheduling and the segmentation of the CAN Message Identifier to provide 8 priorities, as shown in table 1. Lowest priority-number messages are transmitted first.

Priority	Expiry Time
0 (SYNC)	-
1 (HRT)	2ms
2 (HRT)	16ms
3 (HRT)	128ms
4 (SRT)	16ms
5 (SRT)	128ms
6 (SRT)	2048ms
7 (NRT)	∞

Table 1: MilCAN priorities

SYNC and HRT have guaranteed latencies, while HRT and SRT have guaranteed and probable latencies respectively up to their individual Expiry Times. NRT messages are the lowest priority messages and never expire. The Expiry Times are adjustable according to the application [2].

MilCAN Scheduling

The scheduling is controlled using Time Division Multiple Access (TDMA) with a 2ms slot. A Master Node is used to keep the segment synchronized by sending a specific SYNC message every 2ms to signal the beginning of the new Sync-Slot.

At every Sync-Slot a MilCAN node is allowed to send one or more messages with SYNC/HRT priorities, while SRT messages are allowed to take over empty HRT slots and idle bus time. Due to the CANbus prioritized Media Access Control (MAC) protocol, messages transmitted from all the segment nodes are arbitrated and delivered highest priority first. Undelivered HRT/SRT messages are scheduled to be expired and dropped, if their delay (within any software queues or due to high bus traffic) gets higher than their maximum lifetime. NRT messages never expire (they are transmitted on bus-idle times).

Using this scheme, MilCAN can control at Network layer the message flow. Running independently from the user's application it reduces development time, as the developer does not have to implement any monitoring to protect the network from excessive traffic.

Sync Master

In a MilCAN segment a node operates as a Master node that notifies the rest of the nodes when a transmit slot has changed, by sending the Sync frame every 2ms. Every node has the ability to become a Sync Master of its segment. As every node has a specific address (within a 255 range address space), the one with the lowest address becomes the dominant Sync Master. The other nodes passively monitor the Sync frames and in case the current Master fails they arbitrate so that the next lowest address node becomes the new Sync Master.

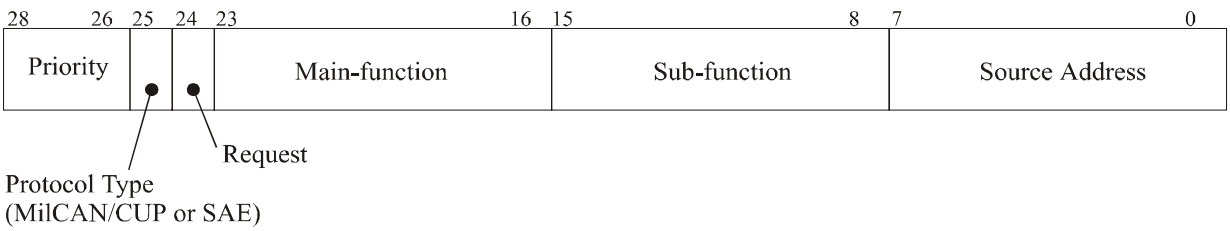

Figure 1 : MilCAN Message Identifier format

VSI Bridge

The VSI Bridge was developed as an initial test-bed for evaluating various networks as high-speed backbone for MilCAN segments. The target was to provide a Standard Network Interface layer where different networks would plug-in and route encapsulated MilCAN frames, while maintaining the MilCAN priorities both within the system and over the network. Routing, filtering and monitoring functionality was added, making the system a Bridge/Router (BRouter). The overall design of the system is shown in figure 2.

The software was developed using wrapper libraries to be able to run on various Operating Systems (Linux, Microsoft Windows, Sun Solaris, etc), including embedded systems. In the current implementation the Bridges run on Linux workstations and embedded network processor.

Bridge Structure

The internal architecture of the Bridge is divided in four main parts, the NAT that does the filtering and routing, the Network Interfaces layer where all the network components are plugged in, the NCC that handles various internal functions and monitoring, and the Global Pool which is used to store and manipulate MilCAN frames.

MilCAN frames enter the Bridge from the Network interfaces and then are forwarded to the NAT where, depending on the configured rules, they are dropped or routed to the destination interface. There is also the ability to modify the CAN frame identifier and mirror it to multiple destination interfaces

The communication between these components is done by a prioritized message-passing interface. Tokens implement the same 8 level priority scheme as MilCAN, reducing latencies on high priority frames.

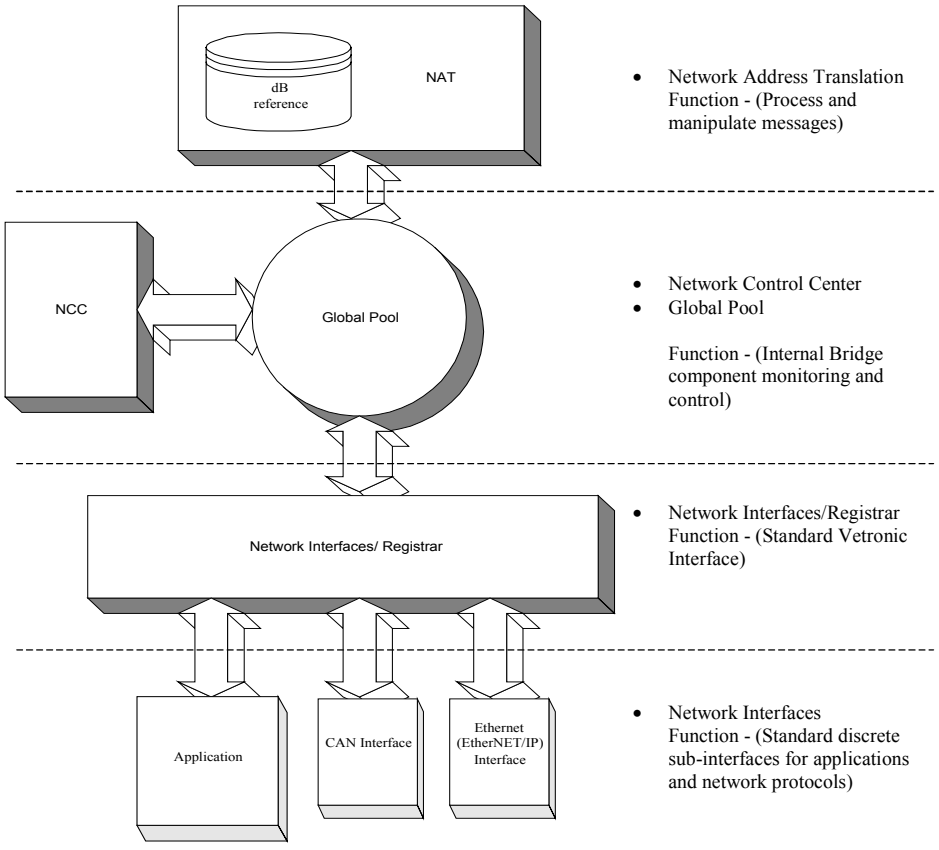

Figure 2 : Bridge component structure

As the Bridge is meant to run on an embedded system where the available resources are limited, emphasis was given on the efficient usage of the CPU cache and memory. Embedded processors have low speed memory busses and small CPU cache, compared to workstation PCs. To maximize the effective processing throughput of the CPU, memory transfers have to be minimal. In the Bridge the use of the message-passing interface combined with the Global Pool component reduce the memory footprint while keeping the system modular.

A Graphical User Interface (GUI) is used to configure the Bridge remotely. Running the GUI on a PC (desktop/laptop) which is attached to the backbone, one can connect to the available Bridge(s) and manipulate the rules stored in the internal NAT databases. Additionally, statistical information about the traffic load of each interface can be obtained, and also real-time monitoring of the traffic being routed through the Bridge.

Bridge Interprocess Communication

The intercommunication within the Bridge is handled by the Mailer, a custom software component that uses shared memory and a message/token based communication. The internal queues use the 8 level priority scheme of MilCAN (SYNC, HRT1-3, SRT1-3, NRT).

Processed MilCAN frames and internal Bridge commands are stored in the shared memory and a token is created for each. These tokens are then forwarded between the main Bridge components for processing, depending on their functionality.

For a token to be transferred a “message” is created which includes the priority of the attached token. In this way critical commands can have higher priority (configuration, notifications, etc) than others (monitoring, logging) and reduce reaction latencies. In addition, messages with CAN frames are tagged with their MilCAN priority, to maintain the MilCAN protocol requirements.

With this message-passing interface an internal network is formed where components can be added and interoperate easily without the need for modifications to the existing ones. Theoretically, the Mailer could be implemented in hardware allowing the whole system to be spread into two or more processors sharing the processing load and thus lowering the system latencies.

NAT

The Network Address Translation (NAT) component handles the routing and filtering of bridged MilCAN frames. An internal database holds the routing rules configured from the GUI. Incoming frames, from the Interfaces, are checked with the filtering rules that are stored in the internal database. If a match is found the relative command is executed. If no match is found the frame is discarded.

The available commands are Route, Drop, Mirror, and Modify. The Route and Drop forward the frame to a specific interface or discard it respectively. A Mirror command creates multiple copies of the frame and forwards it to more than one destination interfaces. With the Modify command one or more fields of the MilCAN frame (priority, message id, source node id, CAN frame flags) can be modified. The rules are executed serially based on their index, until a route or drop command is found.

Network Interfaces

One of the main aspects of the Bridge is the ability to use different types of network protocols transparently to the rest of the system. As the main data unit is a MilCAN frame, a standard interface (the Standard Vetronics Interface) was devised to accommodate all transfers.

The Network Interfaces component consists of a layer that handles the communication with the rest of the Bridge, and the Interfaces that are independent

units implementing the stack of their specific network protocol. These units handle the encapsulation of the Standard Interface frames into their protocol, and their transmission/reception. Incoming frames once unpacked are forwarded to the upper layer, which in turn forwards them to the NAT. After passing through the filter if they have a destination they are sent back to the upper layer, which forwards them to the relative interface.

The combination of the Standard Interface along with the upper layer allows the network interfaces to operate transparently to the whole system, and also be dynamic and hot pluggable. Any number of interfaces, either of identical or other protocols, can co-exist and interoperate as long as they can encapsulate the Standard Interface frames. In the current VSI Bridge implementation the interfaces used were CAN, standard TCP/IP over Ethernet, a custom EtherNet/IP [3] implementation, a command line user interface, and a graphical user interface.

Standard Vetronic Interface

The standard interface structure is shown in figure 3. It consists of 16 bytes that include all the fields of a MilCAN frame. It is used as a standard structure to represent a MilCAN frame within the Bridge. Additional information is also encapsulated, used for the communication with the embedded controllers that handle the MilCAN protocol communications.

In the Bridge implementation the MilCAN code should run independently of the Bridge on a dedicated processor, in order to fulfill the MilCAN timing requirements. This would typically be implemented on a microcontroller with limited processing power. Hence the communication with the Bridge has to be with the smallest overheads possible. Instead of having to implement a custom protocol to encapsulate MilCAN frames along with the commands exchanged with the Bridge, the following additional information were embedded within the standard interface:

Command/Data Frame: distinguishes if the frame is a command (used to send custom commands from/to the MilCAN controller) or a MilCAN frame.

CAN Segment: shows from which CAN port/segment the frame arrived or to which it should be sent (for dual-CAN controllers).

Using the Command/Data Frame field the SVI frame can be converted from a MilCAN frame to a custom structure holding any application based commands, providing flexibility and compatibility to current and future implementations.

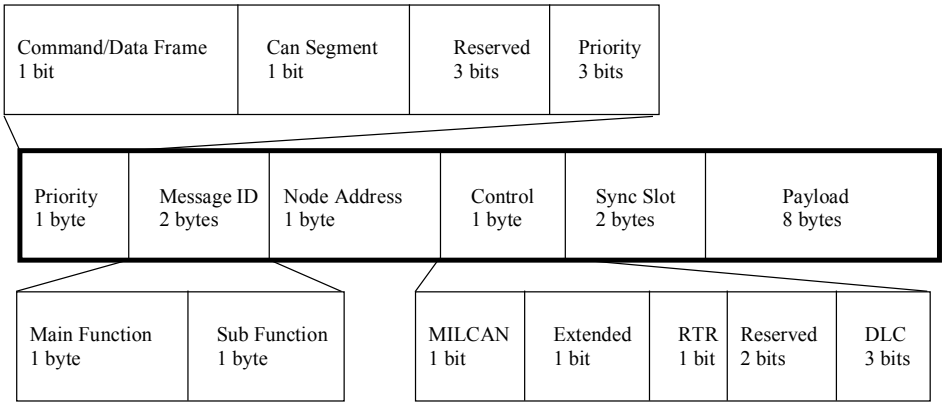

Figure 3 : Standard Network Interface frame format

VSI Rig

The test bed VSI Rig configuration at QinetiQ consists of three MilCAN segments, as shown in figure 4. The Multimedia Control segment, used for audio/video nodes like camera units, the Automotive segment with the engine controller, transmission controller, and driver controls, and the Utilities segment hosting vehicle devices such as lights, wipers etc and drivers instruments. Three Bridges, three Ethernet switches with 100Mbit ports and Gigabit backbone, and a CAN network, form an integrated system network.

MilCAN Interface

For the MilCAN interface a 40MHz 16bit dual CAN Infineon microcontroller with a USB1.1 device controller was used. The MilCAN code running within the node redirected all MilCAN traffic from the nodes to the Bridge through the USB, and vice versa. Both CAN busses were used and the communication was done using Standard Vetronics Interface frames.

Ethernet Interface

Switched Ethernet network was used for the high-speed backbone. The 100Mbit Full Duplex speed, although non-deterministic, provide a low-latency transfer medium for the low traffic of one or two 1Mbit MilCAN segments. Limited only by the TCP/IP stack efficiency and hardware/software checksumming, the average packet latency is around 50ns. Utilizing the Ethernet's MTU of 1500 bytes for block transfers the system processing overheads are reduces even further. As an example implementation, standard TCP/IP sockets were used to transfer MilCAN frames between the Bridges.

EtherNet/IP Interface

Additional to the standard TCP/IP implementation, a custom implementation of EtherNet/IP (Ethernet Networks / Industrial Protocol) was used on top of TCP/IP. This protocol defines specific packet headers, ports, and functions such as remote device enumeration, and time triggered actions. The current VSI Bridge EtherNet/IP interface includes only the basic functionality.

GUI Interface

A Graphical User Interface was developed to configure a Bridge locally or remotely. The user can insert/remove/list the filtering rules, view basic traffic statistics per interface, and also monitor traffic. These functions can be performed on a local Bridge or on a remote one, using either the Ethernet or the EtherNet/IP interfaces.

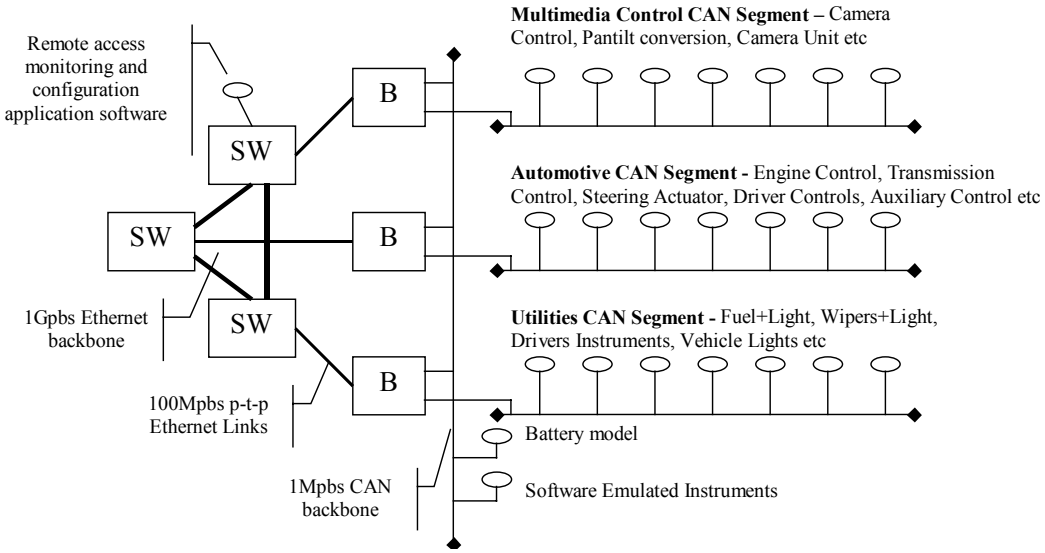

Figure 4 : QinetiQ MilCAN Test bed topology

VSI Bridge performance

To evaluate the performance of the Bridge basic scenarios were simulated on the QinetiQ rig, using the Automotive and Utilities segments. Two Bridges with Ethernet and CAN interfaces acted as a backbone.

The Bridges run on two laptops with 650Mhz PIII processors, and the MilCAN interfaces on two C167CS 16bit 40Mhz dual CAN controllers.

A simulated scenario was run with the Bridges configured to route all the traffic through the CAN backbone segment and through the Ethernet network. The observed traffic per priority was the following (approximate values, also shown in figure 5):

HRT 1: 800 frames/sec
HRT 2: 800 frames/sec
HRT 3: 8 frames/sec
SRT 1: 12 frames/sec
SRT 2: 41frames/sec

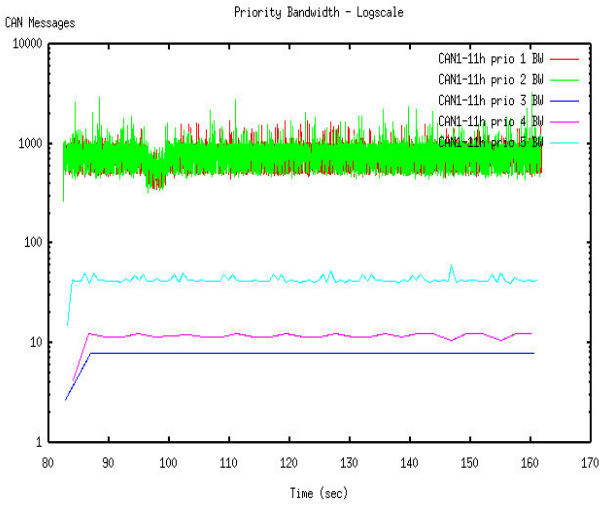

Figure 5 : Bandwidth usage per priority

The observed latencies within one Bridge are shown in figure 6. For HRT1 and HRT2 the average latencies are approximately 325ns and 375ns respectively.

These are initial performance measurements as the Bridge is currently under development. The internal components are being profiled to detect bottlenecks and new database and queue

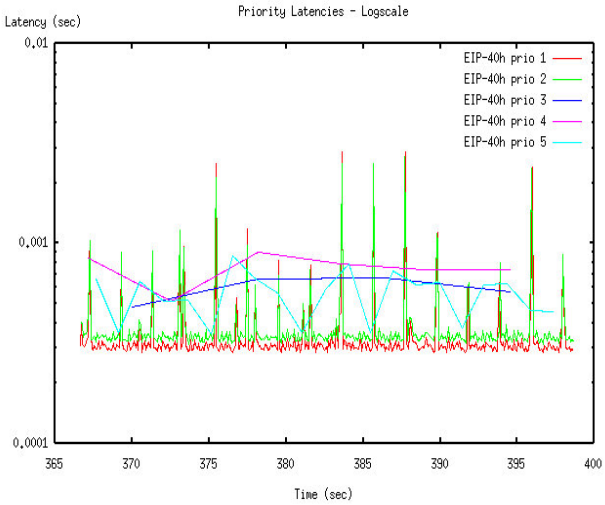

Figure 6 : Message latencies per priority

implementations are being evaluated to decrease message latencies as much as possible, as the system is targeted to run on embedded processors with limited processing power. In parallel the MilCAN Interface Node code is being developed along with the Bridge, to further optimize the system and utilize features like concurrent CAN/Ethernet traffic sharing, with cut-through CAN bridging, and dynamic filtering rules from node message flooding.

Conclusion and Discussion

With the current advances in high speed networking media protocols their use in real-time embedded systems seems feasible using stock hardware products and focusing the development on the overlying software protocols, instead of designing custom networks to fulfill the project requirements.

Ethernet, by breaking the speed barriers of embedded networks by 1000 to 10000 times-fold, along with the wide range of available hardware (switches, routers) and transfer mediums (copper, fiber, wireless), provide a promising choice as a backbone for an embedded network. Many automation micro-controllers ranging from 8bit to 32bit are being developed to include an Ethernet physical layer along with a basic TCP/IP stack. As Ethernet with TCP/IP is used in almost every computer network globally, most software programmers are familiar in working with it, thus providing a wider range of

experienced developers in the IT market. Also the presence of a Video capable network within a military or commercial vehicle allows the use of High Definition cameras (military) or DVD streaming (civilian vehicles).

The VSI Bridge proof of concept implementation shows that it is possible to use Ethernet to relieve an embedded deterministic network running MilCAN by offloading low and non-critical traffic, while also providing a wide range of monitoring and configuration features, that would otherwise not be feasible.

Acknowledgements

The work presented in this paper was funded by the UK MoD as part of the Applied Research Programme and undertaken by QinetiQ and the Communications Research Group of Sussex University.

References.

[1] MilCAN Specifications, http://www.MilCAN.org/

[2] Steven T. Majoewsky, Colin Davies, “MilCAN: Adapting COTS CANbus to Military Vetronics”, 8th International CAN Conference, 2002, USA.

[3] Open DeviceNet Vendor Association,”EtherNet/IP”, http://www.odva.org/

Periklis Charchalakis
University of Sussex
Falmer
Brighton
East Sussex
United Kingdom
BN1 9QT
Tel: +44 (0) 1273 678957
Fax: +44 (0) 1273 678399
P.Charchalakis@sussex.ac.uk
www.comms.engg.susx.ac.uk

George Valsamakis
University of Sussex
Falmer
Brighton
East Sussex
United Kingdom
BN1 9QT
Tel: +44 (0) 1273 678957
Fax: +44 (0) 1273 678399
G.Valsamakis@sussex.ac.uk
www.comms.engg.susx.ac.uk

Bob Connor
QinetiQ
Cody Technology Park
Ively Road
Farnborough
United Kingdom
GU14 0LX
Tel: +44 (0) 1252 397011
Fax: +44 (0) 1252 397011
Rmconnor@QinetiQ.com
www.QinetiQ.com

Dr Elias Stipidis
University of Sussex
Falmer
Brighton
East Sussex
United Kingdom
BN1 9QT
Tel: +44 (0) 1273 678957
Fax: +44 (0) 1273 678399
E.Stipidis@sussex.ac.uk
www.comms.engg.susx.ac.uk