INCA: Support for IN Using the Tempest

Rebecca Isaacs and Richard Mortier*
{Rebecca.Isaacs,Richard.Mortier}@cl.cam.ac.uk
University of Cambridge Computer Laboratory, UK

Abstract

The provision of true multi-service networks has been hampered by inflexibility in network control software, despite being both desirable and technically feasible. The Tempest is a framework that alleviates this problem by using devolved control mechanisms and strict resource partitioning to allow many different control systems to run simultaneously on one physical network.

We explore the deployment of telephony services within the Tempest environment through an Intelligent Network (IN) control architecture (CA)—INCA. INCA is a control system that mimics a traditional IN architecture, while taking advantage of the flexibility of the Tempest to offer value-added IN services.

1 Introduction

Although recognised as desirable, no true multi-service networks currently exist. Their deployment has been hampered by inflexibility in network control systems, which are often targeted at just one type of service and not amenable to adaptation for others. Even when control systems are designed from the outset to cater for multiple services, shortcomings are revealed when unanticipated network services and applications are devised. In particular, simply adding more functions to such a system results in large, unwieldy complexity where maintenance and upgrades are costly and error-prone.

The Tempest is a framework that was conceived with the aim of alleviating many of these problems. It uses devolved control mechanisms together with strict resource partitioning to allow many different control systems to run simultaneously on the same physical network. For example, an instance of ATM UNI/NNI signalling can be active at the same time as instances of MPLS and RSVP, each operating in isolation and unaware of the presence of the others.

*Supported by an EPSRC CASE award, in collaboration with BT.

The ability to run multiple network control systems at once has obvious advantages for development and upgrade activities. Furthermore, it has consequences for the attributes of these controllers—any given network control system does not now have to be general-purpose and monolithic; in some situations it may be preferable to use *service-specific* control software tuned to the resource requirements and behaviour of a particular network service.

In the work presented in this paper, we have explored provision of a telephony service within the Tempest environment through the development of an Intelligent Network control architecture—INCA. INCA is an example of a service-specific control system that mimics a traditional IN architecture, while also taking advantage of the flexibility of the Tempest to offer value-added IN services.

In Section 2 we give an overview of the Tempest, and then present the design and implementation of INCA in Section 3. Some related work is described in Section 4.

2 The Tempest

In this section we briefly describe the components that make up the Tempest, and then discuss the use of dynamically loadable control, a mechanism employed within INCA. Further detail on the Tempest can be found in [1] and [2].

2.1 Core Components

Within the Tempest, control of the physical network is, as far as possible, devolved from the internal network elements to general-purpose workstations. In addition, the resources of the network are partitioned, and the resulting resource partitions are allocated to *virtual networks*. The management and control software of a virtual network is known generically as its *control architecture*.

The partitioning of the network is performed by subdividing the resources of individual switches into one or

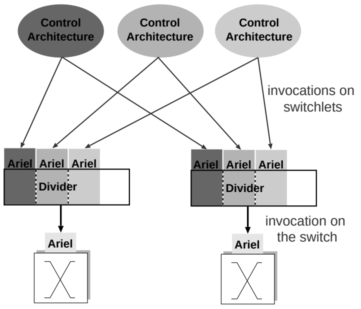

Figure 1: CAs operating simultaneously.

more logically separate parts known as *switchlets* [3]. Each of these is presented to its owning control architecture through an open switch control interface, which gives the illusion that the control architecture is managing a single switch in its entirety. This is the crucial feature that allows multiple control architectures to coexist on a single physical network, while also giving them fine-grained control of the resources they have been allocated. Examples of open switch control interfaces include GSMP [4] and VSI [5]. The Tempest uses an interface developed in the Computer Laboratory called Ariel [1, 2].

Invocations on switchlet interfaces are policed on the control path by a Divider (one per switch) to ensure there is no interference between control architectures. Figure 1 depicts multiple control architectures sharing the resources of the physical network.

An entity called the Network Builder is responsible for the creation and deletion of virtual networks, and for allocating the resources required by those virtual networks. A control architecture acquires a virtual network by asking the Network Builder for a new network comprising some specified resources. The Network Builder locates the Divider at each constituent node, and requests from it a new switchlet. After creating a switchlet the Divider returns a new Ariel interface for that switchlet to the Network Builder. The switchlet interfaces are passed back to the control architecture, which can then make invocations on switchlets directly and hence control its partition of the physical network resources.

2.2 Dynamically Loadable Control

In an environment with devolved control, it is feasible to extend or modify an active control architecture by dynamically loading code into it. This process does not necessarily disturb either the normal operation of the control architecture, or any of its active connections. This is similar to *active networks* [6], discussed in Section 4.1.

Two methods of dynamically loadable control have been proposed for the Tempest:

- Code is loaded into a general-purpose control architecture and then executed only in association with a predetermined set of network resources. These resources typically form a connection, with the code invoked whenever such a connection is created or deleted; hence the name *Connection Closure* [7] to describe this mechanism.

- The dynamically loaded code is not limited to a given subset of the control architecture's resources, but can affect the operation of the control architecture itself, for example by augmenting its interface, or by overriding internal methods. This form of dynamically loadable control is referred to as *Elastic Control* [8].

We make use of the second form of dynamic control within INCA to enable the easy introduction of new IN services. The exploitation of this technique also allows us to offer services that are not currently feasible in traditional IN architectures as described in Section 3.2.1.

3 INCA

INCA—Intelligent Network Control Architecture—is a lightweight control architecture that provides facility for IN within the Tempest environment. The motivation for INCA is twofold: to realise a telephony control architecture on a true multi-service network, and to explore the benefits that can be obtained using dynamic control techniques in a realistic example.

3.1 Design

Figure 2 illustrates the entities within INCA and shows the steps involved in establishing a basic two-party call (i.e. not invoking any IN services). The underlying network, which is actually a *virtual network*, is represented by three switches, with endpoints—telephones—attached to the two edge switches. The components shown are as follows:

Phone Manager An instance of the Phone Manager runs for each switch to which phones are directly attached, and is configured to accept events that occur at its dependent phones. It instantiates a call state model, augmented with IN service detection points (*triggers*), for each phone, and also keeps track of their physical endpoint addresses. In the establishment of a basic call, after a valid number

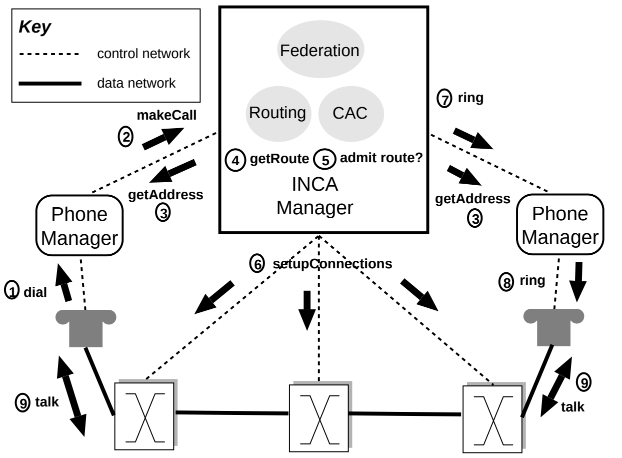

Figure 2: Establishing a two-party call in INCA.

has been dialled at the handset **(1)**, the Phone Manager will request the INCA Manager to set up a call from source to destination **(2)**.

INCA Manager The INCA Manager is the core of the control architecture. It creates and destroys connections across the network through the Ariel interfaces available to it, and it interacts with the CAC, Routing and Federation modules. After being contacted by a Phone Manager and asked to set up a call connection, the INCA Manager finds out the physical endpoint addresses of the parties involved **(3)**. It then determines a route between those addresses **(4)**, and checks admissibility of the call **(5)**. Steps **(4)** and **(5)** may be repeated if necessary. If all is well the INCA Manager sets up the physical connections from end to end via its Ariel interfaces **(6)**, and informs the destination's Phone Manager that a connection request has been made **(7)**. It is the responsibility of that Phone Manager to contact the appropriate handset **(8)**, and if the ring is answered then data transfer can commence between the endpoints **(9)**. The INCA Manager does not maintain any state about the progress of this call—if a ring is not answered by the called party, the Phone Manager of the calling party must inform the INCA Manager in order for the resources to be released. This simplifies the distribution and replication of components of the INCA Manager.

CAC Module Within the Tempest, *Call Admission Control (CAC)* operates on two levels. The Divider polices invocations on switchlets to ensure that a virtual network cannot exceed its resource allocation. This level of CAC should give a result that is the same as or more conservative than CAC on the switches themselves. A control architecture may also make use of additional CAC tailored to its own requirements; for example if a particular route is oversubscribed it may be possible to use an alternative route.

Routing Module End-to-end routing takes place in conjunction with the CAC Module and the Federation Module. INCA has complete knowledge of the topology of its virtual network, which is used in making routing decisions. Routing may also be influenced by congestion or by sophisticated IN service logic.

Federation Module Federation handles call setup between INCA Managers in different Tempest domains. The federation of virtual networks, possibly across non-cooperating intervening networks, is an issue being addressed in ongoing work.

Although these components can be thought of as single, centralised entities, it is quite feasible that their implementation is distributed and replicated to provide fault tolerance, high availability and scalable performance.

3.1.1 Invocation of IN Services

IN services can be invoked either centrally by the INCA Manager, or at the endpoints of a call by the local Phone Manager. Invoking an IN service consists of executing a script; such scripts can be updated dynamically, allowing introduction of new services or modification of existing ones without disruption to existing calls.

A Phone Manager will have an executable script corresponding to each trigger detection point in its call state model. In the extreme case, the call state model can itself be defined inside a dynamically loadable script, hence facilitating the co-existence of non-standard and novel types of call with existing calls, without forcing conformance to an inappropriate call model. Similarly, the INCA Manager checks for the existence of dynamically loaded IN service scripts at certain fixed points of the call state model throughout its operation.

The simplest way of disseminating IN scripts is to physically place a copy of the script in a well-known location accessible to, and preferably also local to, either the Phone Manager or the INCA Manager. This is the mechanism we have used in our initial implementation. Other means of carrying out this task can be envisaged, for example, using mobile agents to transport scripts "on-demand" [9].

3.2 Implementation

INCA is implemented in a scripting language called Python over the existing Tempest codebase, which is written mostly in C++, and consists of the core components described in Section 2.1 together with a graphical visualisation tool and a network management interface. The physical network comprises 5 ATM switches, 4 audio/video codecs, and 4 Sun workstations.

3.2.1 IN Services

In order to assess our approach we have implemented three simple IN services in INCA. The first is straightforward call forwarding which is a ubiquitous IN service in which the service logic is located centrally. The second service is a variation on call forwarding where the IN processing is best done locally to the calling party. The third service—advance resource reservation—is more difficult to offer in current IN architectures, but straightforward within INCA.

The standard call forwarding service logic is located within the central INCA Manager. In this case, before calling `getRoute()`, a special destination area code is remapped if it comes from a particular switch. The destination number is replaced, and the new number used in all subsequent steps performed in placing and removing the call.

In some cases it is better to place the intelligence as close to the specified callers as possible—for instance in order to translate only for certain originating phone numbers within a given area code. We have implemented this service in a manner very similar to the above, but by placing the scripts at the appropriate Phone Managers. After the number has been dialled, but before the INCA Manager is contacted, the script is executed, remapping the destination number.

The third sample IN service we provide in INCA is advance resource reservation, where parties can reserve a connection for a particular future time and duration. This service involves the invocation of a script at the INCA Manager that suspends the call until the specified time, together with a script in the CAC Module that ensures resources are guaranteed to be available at the appropriate time.

3.2.2 Scalability

There are two main areas of concern regarding scalability: firstly, the architecture may span a large geographical area, and secondly, it must be able to handle large numbers of calls and call requests concurrently.

We have measured the overhead of invoking IN services in INCA by comparing the overhead of the three different types of IN services described above in Section 3.2.1 to that of setting up a basic two-party call over 3 switches. The elapsed time for each operation is between 500 and 600 ms while the CPU consumption for each was approximately 66 ms[1]. Given that this is a rudimentary prototype we believe these figures are encouraging. Faster set-up times can easily be achieved by optimisations within the INCA Manager. For example, connections could be cached to minimise route calculation overheads, or could be set up in parallel by making invocations on multiple switchlets at the same time.

The size of the geographical area managed by an INCA instance has implications for connection establishment times for basic calls, and for the delay caused by invocation of an IN service. The latter delay is somewhat mitigated by the facility within INCA to position IN service logic close to the requesting endpoint, i.e. in its local Phone Manager when appropriate. The physical separation of the signalling (SS7) network and the data network in telecommunications networks has a number of advantages for overall scalability and security of the network. Whilst the signalling network is not a distinct network in INCA, this same separation is achieved by acquiring two virtual networks and exploiting the resource partitioning provided by the Tempest to ensure that sufficient resource is always available for signalling[2]. It should be noted that use of SS7, or any other standard signalling protocol within INCA is not precluded.

4 Related Work

4.1 Programmable Networks

Networks in which programs can be inserted into the routers or switches and then executed on the messages passing through those nodes are known as *active networks* [6]. This network programmability has many of the benefits of the Tempest, such as flexibility and potential for customisation, but differs in that the control and data paths are not distinguished. This makes it

[1] We recognise that the devolving of control from the switches may give rise to concerns about the performance of control operations. Connection setup times, including inter-entity communication within the control architecture, have been measured to be in the region of 11 ms on a single switch [10].

[2] If desired, the two networks can even have different physical topologies, which has consequent advantages for incorporating redundancy and high availability in the signalling network without paying over the odds for unused resources in the data network.

more difficult to provide hard guarantees for differentiation of services.

An IN architecture realised using active network concepts is described in [9]. It uses mobile agents to migrate service logic to the most appropriate location in the network. This location is determined mainly by whether the service has been requested by the calling party, the called party or by both. Call processing is suspended while the mobile agent makes its way to the appropriate switch, which must be modified to be able to host mobile agents. This mechanism could be adopted inside INCA as a means of disseminating new scripts to Phone Managers, but INCA is more general in that nodes do not *need* to be able to support mobile agents, and the switches themselves remain entirely unmodified.

4.2 Broadband IN

INSIGNIA [11] considers the use of the INA over broadband ISDN as a mechanism for introducing multimedia services into such networks, without the need for additional complexity in the network protocols. A model is proposed that integrates IN with standard ATM protocols, and the results of a trial spanning three European countries are presented. Although promising in the short term, as it is entirely based on existing and standardised signalling protocols, INSIGNIA requires enhancement of ATM switches to also act as SSPs. In the long term it is limited by its assumption of UNI signalling in the network.

IBIS [12] extends the INSIGNIA work by replacing the traditional IN components with TINA service components. IBIS is suggested as an evolutionary step in the difficult task of migrating intelligent broadband networks to the TINA architecture. However, since all the IN components must be TINA-compliant, the proposed INA is irrevocably tied to TINA.

5 Summary

Monolithic, inflexible and complex control software has hampered the deployment of multi-service networks. The Tempest provides a solution by allowing many, possibly different, controllers to co-exist on a single physical network, each with exclusive access to a subset of the available resource.

The Tempest allows us to build a true multi-service network—we have demonstrated this by implementing a rudimentary telephony service that runs concurrently with other control architectures developed in previous work, including Q.2931 and a control architecture optimised for video-conferencing. The increased flexibility gained from devolved control has been exploited by incorporating an IN service into INCA that would not be possible using current INAs, namely advance resource reservation. Although currently an immature technology, the Tempest is very promising as a solution to the difficulties inherent in multi-service network control.

References

[1] Sean Rooney, Jacobus E. van der Merwe, Simon A. Crosby, and Ian M. Leslie. The Tempest: A framework for safe, resource-assured programmable networks. *IEEE Communications Magazine*, 36(10):42–53, October 1998.

[2] Jacobus E. van der Merwe, Sean Rooney, Ian Leslie, and Simon Crosby. The Tempest—a practical framework for network programmability. *IEEE Network Magazine*, 12(3):20–28, May 1998.

[3] Jacobus E. van der Merwe and Ian Leslie. Switchlets and dynamic virtual ATM networks. In Aurel Lazar, Roberto Saracco, and Rolf Stadler, editors, *Integrated Network Management V*, pages 355–368. IFIP & IEEE, Chapman & Hall, May 1997.

[4] P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching, T. Lyon, and G. Minshall. Ipsilon's General Switch Management Protocol specification version 2.0. RFC 2297, March 1998.

[5] William P. Buckley. Virtual Switch Interface (VSI) implementation agreement. Available from `http://www.msforum.org/`, November 1998.

[6] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[7] Sean Rooney. Connection closures: Adding application-defined behaviour to network connections. *Computer Communication Review*, 27(2):74–88, April 1997.

[8] Herbert Bos. Elastic network control with future reservations. In *Proceedings of the 4th European Research Seminar on Advances in Distributed Systems (ERSADS)*, Madeira, Portugal, April 1999.

[9] Markus Breugst and Thomas Magedanz. Mobile agents—Enabling technology for active intelligent network implementation. *IEEE Network Magazine*, 12(3):53–60, May 1998.

[10] Sean Rooney. *The Structure of Open ATM Control Architectures*. PhD thesis, Cambridge University, Computer Laboratory, UK, February 1998.

[11] George N. Prezerakos, Stefano Salsano, Alexander W. van der Vekens, and Fabrizio Zizza. INSIGNIA: A pan-European trial for the intelligent broadband network architecture. *IEEE Communications Magazine*, 36(6):68–76, June 1998.

[12] Marco Listanti and Stefano Salsano. IBIS: A testbed for the evolution of intelligent broadband networks toward TINA. *IEEE Communications Magazine*, 36(6):78–91, June 1998.