Component Coordination: A Telecommunication Case Study

Pamela Zave, Healfdene H. Goguen, and Thomas M. Smith
AT&T Laboratories, Florham Park, New Jersey, USA
{pamela,hhg,tsmith}@research.att.com

26 August 2003

Abstract

Personal mobility is probably the most desired of all telecommunication capabilities, but it presents many design difficulties. This paper reports on the development of a voice-over-IP service centered on personal mobility. It explains the functions of personal mobility, as well as its functional dependences and interactions with other major features. It shows how configurations and feature interactions can be managed with the help of a component architecture. The presentation emphasizes two major component-coordination mechanisms.

1 Introduction

This paper describes how a component architecture and its component-coordination mechanisms were used to build a complex voice-over-IP (VoIP) service. The service is centered on personal mobility, although it offers several other features that are not directly related to it. The service is unique, and was developed with surprising speed—about two months to a working version.

The paper is an interweaving of two themes, both of which are introduced in Section 2. The first theme is the ramifications of personal mobility in telecommunications. Although personal mobility is known to be important, it has many more aspects, interactions, and other subtleties than are generally recognized. The paper elucidates the problems and offers solutions to them.

The second theme is the component-coordination mechanisms offered by Distributed Feature Composition (DFC) [12, 13], a component-based architecture for telecommunication services. One of the two major mechanisms is novel, and the other is used in an unusual way. Together they are a powerful tool for supporting modularity, providing code re-use, and managing complex feature interactions.

The two themes are interwoven in the paper because the architecture is not just a solution to the problems of personal mobility. The structure and modularity provided by the architecture *make it possible* to understand personal mobility and its effect on other features well enough to see what the real problems are. Therefore we first present all feature functions as implemented in the component architecture (Section 3), and then go on to analyze and manage their interactions within the architecture (Sections 4 and 5).

This is followed by a brief discussion of our implementation experience (Section 6). The paper concludes with lessons learned, related work, and evaluation (Section 7).

2 Overview of the two themes

2.1 Personal mobility

Personal mobility is probably the most desired of all telecommunication capabilities. In addition, people want a variety of other features that are "personal" in two senses: they are personalized in their functions, and the owner can use them wherever he happens to be.

Cellphones are popular because they provide personal mobility. This kind of mobility is called *device* or *network-based* mobility because a physical device is moving without changing its network address. Cellphones have many limitations with respect to voice quality, battery life, geography, and the user interface. Because of issues such as billing, a person cannot always use the same device for all his work and personal needs.

Clearly device mobility is not a comprehensive solution to the problem of personal mobility. It must be used in cooperation with *software-based* mobility,

in which a personal address is mapped by software to the address of a device. The device is one among many; the choice of device is based on data describing the subscriber's current situation.

At best, providing software-based mobility would be a big job. Telecommunication software is notoriously difficult to develop. It is complex, long-lived, distributed, concurrent, and real-time. Features are being added all the time, and they have many interactions [5]. The problem of feature interaction is so difficult that it has engendered its own series of conferences [1, 3, 4, 7, 9, 14].

The challenges of personal mobility go far beyond the obvious. First, there are little-understood functions and issues. For example, for a person to be able to access his personal telecommunication environment from any telecommunication device, personal mobility must apply to calls that he places as well as calls he receives. If a personal environment is accessible from any device, it must be possible to authenticate the user. If a person can make a personal call from any device, he should be able to change devices mid-call, since the device is not an intrinsic attribute of the communication. Other issues include the privacy of users and the returnability of calls.

Second, features for multiparty control, performing functions such as switching, conferencing, and transfer, are also popular and highly valuable. If such a feature is available, is it associated with a device or with a person? The answer to this question has a profound effect on its interaction with personal mobility. Each answer has important advantages and disadvantages, yet the question is seldom asked.

Third, the more features available to a user, the more elaborate a user interface is needed to control them. If the features are personal and accessible from any device, they must be controllable from any device. Yet devices vary widely in their signaling capabilities, and many common ones ("black phones") are impoverished.

Early predictions about VoIP foretold that it would not suffer from the signaling and user-interface limitations of the Public Switched Telephone Network (PSTN). Of course these predictions were wrong, if for no other reason than that the vast majority of VoIP calls today have a PSTN telephone at one or both ends. Currently vendors of VoIP equipment are struggling to support DTMF signaling, which was omitted from the early VoIP standards.

It is clear that voice-based signaling (announcements and DTMF tones), while necessary, is not sufficient. It cannot provide a good user interface to a complex feature such as multiparty control. For this reason, we designed and implemented an alternative approach to augmenting the user interface of a PSTN telephone or other simple voice device.

2.2 The DFC architecture

In DFC a request for telecommunication service is satisfied by a *usage*, which is a dynamically assembled graph of *boxes* and *internal calls*. A *box* is a concurrent process providing either interface functions (an *interface box*) or feature functions (a *feature box*). An *internal call* is a featureless, point-to-point connection with a two-way signaling channel and any number of media channels. In our case study each internal call has exactly one voice channel, and no other media.

Figure 1, like all other figures in this paper, depicts a usage. Boxes are represented as squares, and internal calls are arrows.

The fundamental idea of DFC is pipes-and-filters modularity [21]. Each feature box behaves transparently when its functions are not needed. Each feature box has the autonomy to carry out its functions when they are called for: it can place, receive, or tear down internal calls, it can generate, absorb, or propagate signals traveling on the signaling channels of the internal calls, and it can process or transmit media streams traveling on the media channels of the internal calls. A feature box interacts with other features only through its internal calls, yet does not know what is at the far ends of its internal calls. Thus each feature box is largely context-independent; feature boxes can easily be added, deleted, and changed.

Each internal call is routed to a box by a DFC router running the DFC routing algorithm, so the routing algorithm determines the boxes in each usage. To explain what a DFC router does, we shall use the concept of a *personal call*, defined as an attempt by one person to create a connection to another person. A personal call usually corresponds to many internal calls; we use *call* only when there is no confusion between the two.

An *address* represents a telecommunication device, a person, a group, or some other virtual entity.

A *feature box type* is the name of a box program, which is a program that defines the behavior of a box of that type. Any address can *subscribe* to feature box types. Feature box types subscribed to by an address *in the source zone* are those relevant to a personal call whose source is that address. Feature box types subscribed to by an address *in the target zone* are those relevant to a personal call whose target is that address.

Figure 1 illustrates a simple personal call. It is initiated by a *device interface (DI)* box. It has a single

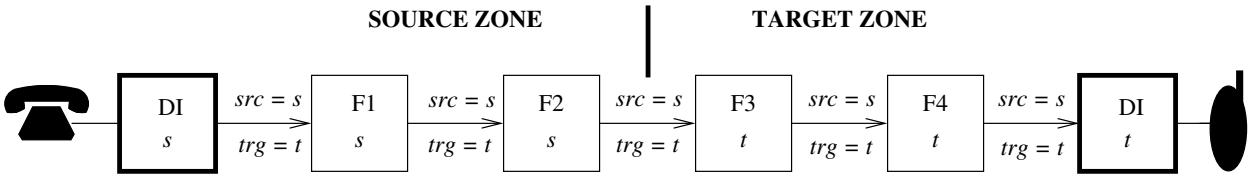

Figure 1: A DFC usage carrying out a single personal call.

source address *s*, which is the address of the initiating device. It has a single target address *t*, which is the address of the terminating device. Between the two *DI* boxes there is a linear chain of feature boxes and internal calls. First there is a *source zone*, consisting of one instance of each of the two feature box types *F1* and *F2* subscribed to in the source zone by *s*. It is followed by a *target zone*, consisting of one instance of each of the two feature box types *F3* and *F4* subscribed to in the target zone by *t*. In each zone, the order of feature boxes is constrained by a *precedence* partial order on feature box types.

In the development of this chain, each feature box received an incoming internal call routed to it by a DFC router. It applied the *continue* method to the setup signal of the incoming call, and used the resulting setup signal to place an outgoing internal call. The setup signal of the outgoing call went to a DFC router, which determined the next feature box type in the chain, and routed the internal call to an instance of that type.

In all the figures, each internal call is labeled with its source and target addresses; the direction of the arrow is the direction in which it was set up. Each feature box is labeled with its box type and the address on whose behalf it appears in the usage.

Most usages are more complex than Figure 1. One complication is that, in applying the *continue* method to a setup signal, a feature box can specify a change of source or target address. If a source-zone feature box changes the source address from *s1* to *s2*, the router will forget all remaining feature boxes of *s1*, and start routing to the feature boxes in the source zone of *s2*. If a target-zone feature box changes the target address from *t1* to *t2*, the router will forget all remaining feature boxes of *t1*, and start routing to the feature boxes in the target zone of *t2*. As a result, a personal call has a *source region* and a *target region*. Each region can consist of several zones, each containing feature boxes subscribed to by a different address.

The state of a chain is maintained within its setup signals, so DFC routers need have no information about particular usages. Part of the state, for example, is a *region* field indicating whether a setup is part of the source or target region. The *continue* method copies the *region* field from old to new setup. If a DFC router receives a setup with *region* = *source* and finds that the source region is exhausted, it changes the *region* of the setup to *target* before sending it to the first box of the target region.

Another complication found in usages is that a feature box type can be *free* or *bound*. If it is free, an internal call destined for an instance of the type is routed to a new, interchangeable copy. If it is bound, each instance of the type is persistent and uniquely associated with an address. Bound boxes enable joins in usages—a join occurs when an internal call is routed to a bound box that is already participating in other internal calls. In the figures, interface boxes and bound feature boxes are drawn with heavier lines than free feature boxes, to indicate their persistence.

Another complication is the *reverse* routing method. To avoid overburdening the reader, we shall introduce this method in Section 3.1.

Some of the signals traveling on signaling channels are built-in and shared by many feature boxes. Most prominent of these are the signals used to indicate the *outcome* of a personal call: *unknown* if the target address is unknown, *avail* if the personal call is successful in reaching a person, and *unavail* otherwise. In a simple usage, outcome signals travel upstream, from the callee end to the caller end.

Another important category of signals is *user-interface* (UI) signals. Some UI signals are sent from feature boxes to devices to display status information and offer command choices. Other UI signals are sent from devices to feature boxes, encoding commands. In a simple usage, UI signals travel between a feature box and the device interface of the feature's subscriber. Each feature box engages in a private dialogue with its subscriber; the signals pass transparently through other feature boxes.

Feature boxes can access persistent *operational data*. Operational data is written both by feature boxes and by Web services. Operational data is usually partitioned by feature so that it does not serve as a covert channel for feature interactions, and is usu-

ally partitioned by address or customer for security.

3 Features

In this paper, a *feature* is an increment of functionality with a cohesive purpose. A complex feature may entail several separate functions. Its implementation may be decomposed into several feature box types, each type implementing one or more feature functions. It may also have dedicated, persistent operational data.

The features in this paper have many behavioral variations: some functions are optional, and some can be performed in different ways. In this paper, for simplicity of presentation, we usually assume that there is a single fixed version of each feature in which all functions are present. Only the most important variations are mentioned.

Most multi-function features can be decomposed into feature boxes in several ways. Designers often need this freedom. A finer-grained decomposition allows simpler and more reusable components; the feature is easier to customize by changing components. Also, many necessary interactions among functions are provided by the architectural infrastructure, and need not be coded. If a necessary interaction among functions is not well supported by the architecture, on the other hand, it may be easier to hide it within a component, leading to a coarser decomposition. Either a fine or coarse decomposition may best accommodate particular legacy components, if their use is required.

Unfortunately, exploration of this design freedom would complicate the paper too much. In this paper we use only the coarsest possible decompositions, with the fewest feature boxes, leading to the simplest usages.

3.1 Personal mobility

The *Personal Mobility* feature allows a person to have a *personal address* that can be used to call him no matter where he is. The personal address is not permanently associated with any device.

Figure 2 shows the two box types of this feature. The voice device with address *v1* subscribes to *ident* in the source zone. Personal addresses *p1* and *p2* both subscribe to *mobil* in both the source and target zones; since Figure 2 depicts a personal call from *p1* to *p2*, it has the source *mobil* box of *p1* and the target *mobil* box of *p2*. Personal mobility includes five functions implemented in these two box types.

The first and most obvious function is *location*. In Figure 2, a caller is trying to reach person *p2*. On receiving an incoming internal call, the *mobil* box in the target zone of *p2* uses operational data to determine that person *p2* is using device *v2*, and then places an outgoing internal call to *v2*. Location functions have many possible behaviors. For example, they can try several possible locations, sequentially or in parallel.

The dual of location is *identification*, which is performed by the *ident* box. This function, which may be performed unconditionally or only when the caller requests it, changes the source address from *v1* to *p1*. This has two advantages: the callee will know that he is being called by *p1*, and the usage will be routed through the source-zone features of *p1*, so the caller will have access to all his personal features and data.

Identification is easily abused. An unauthorized person can use device *v1*, even though it is meant only for the use of person *p1*. Or the owner of *v1* can simply program its features to identify calls as coming from *p1*! For this reason there is an *authentication* function in the *mobil* box.

When a *mobil* box is in the source zone, the authentication function is activated when the box receives its incoming call. It demands a password, or does whatever else is necessary to ensure that the caller is actually person *p1*. Since the routing algorithm guarantees that any personal call with source address *p1* is routed through the *mobil* box of *p1* in the source region, this security measure cannot be bypassed.

Although it is less common, the *mobil* box can also include a function to authenticate the callee, in case an unauthorized person or machine answers the telephone. This form of authentication is activated only in a target-zone *mobil* box, and only when the call is answered. It does not connect caller and callee until a person has proved himself.

Figure 2 illustrates the compositional nature of DFC: each internal-call arrow has dots (indicating ellipsis) in it, emphasizing that while it might be a single internal call, it might also be a subgraph of boxes and internal calls.

The ellipses marked **A** and **B** are of special interest with respect to authentication. Because the precedence order places *mobil* first in the source zone and last in the target zone, all other personal source-zone feature boxes of *p1* will be found in **A**, and all other personal target-zone feature boxes of *p2* will be found in **B**. In each zone *mobil* stands guard between the user and personal feature boxes, so that personal features and data cannot be accessed until the user is authenticated.

The third function of a *mobil* box is an *update* function, so that a user can update his current location from a telecommunication device.

Once the parties are talking, a person may need

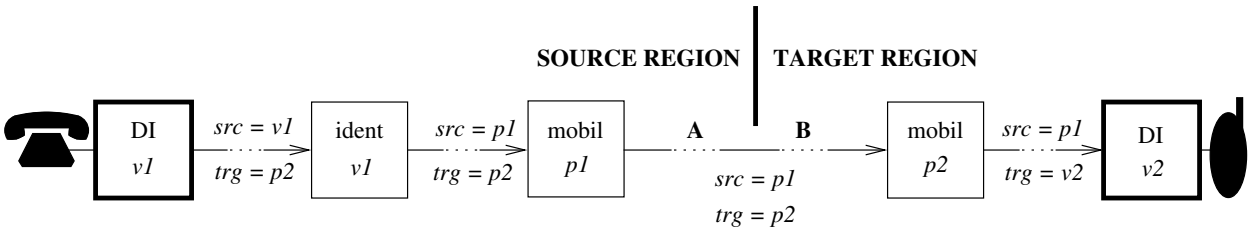

Figure 2: The Personal Mobility feature.

to change his device location while continuing the conversation. For example, person *p1* is using home telephone *v1*, and needs to leave for work. He triggers the *mid-call move* function in his *mobil* box, which places a new internal call to cellphone *v3*. Figure 3 shows this half of the usage after *mobil* has placed a call to *v3*. Then the user answers the call at *v3* and hangs up *v1*. *DI v1* will tear down its internal call, and the teardown will propagate through the usage until it reaches *mobil p1*. When the usage stabilizes, *mobil* will once again be participating in exactly two internal calls.

Of the four functions performed by a *mobil* box, update and mid-call move work exactly the same regardless of whether the box is in the source or target zone of its personal address, and authentication works almost the same in both zones. A personal address subscribes to the same *mobil* box in both source and target zones so that the user will have the same feature functions regardless of whether he is the caller or the callee. A box type that must be subscribed to by an address in both zones is a *reversible* box type.

Recall that a feature box applies the *continue* method to the setup signal of an incoming call, and uses the resulting setup signal to place an outgoing call in the same region. The vast majority of internal calls are placed using the *continue* method, but a reversible box is sometimes required to do something different. Sometimes it must reverse the direction of a chain of internal calls; to accomplish this, it applies the *reverse* method to the setup signal of an outgoing call, and uses the resulting setup signal to place an outgoing call in the opposite region. The *reverse* method also reverses the source and target addresses.

Figure 3 is the result of reverse routing, which is necessary because person *p1* originally *placed* a personal call from a device, and now wants to *be called* at another device. The *mobil* box obtained the setup signal for its third internal call by reversing the setup signal of its outgoing call with target address *p2*, which has *region = source*. The setup signal of the third internal call has *region = target* and a source address of *p2*.

As with the *continue* method, a box program applying the *reverse* method can specify a change of source or target address. In this case the *mobil* box changed the target address from *p1* to *v3*. In doing so, it performed a location function, now locating *p1* at *v3*. Here a *mobil* box originally routed to in the source region is performing the target-region function of location, and placing a call in the target region, which shows just how reversible this box really is.

Although the location function required for personal mobility is widely understood, the identification, authentication, mid-call move, and update functions seem relatively obscure. In particular, the usual way for a person to use device-independent features on an outgoing personal call is to dial a toll-free number to reach a service that will allow him to identify and authenticate himself. Then he must dial a second time to supply the address of his true target. Our identification function allows a user to achieve the same goal without the inconvenient double dialing, when calling from a device whose address subscribes to the right features.

3.2 Multiparty control

The *Multiparty Control* feature allows a person to control multiple personal calls. The person can switch among them, create conferences spontaneously, and perform various transfer functions. Figure 4 shows the boxes of this feature. They are subscribed to by the voice-device addresses *v1* and *v2*, so that these devices now have multiparty capabilities.

The *switching and spontaneous conferencing (SSC)* box type is both bound and reversible. The *switching* function is like an unrestricted version of call waiting. In the figure, *v1* was first used to place a personal call to *v2*. This personal call was routed through *SSC v1* in the source zone, passed transparently through it, was routed through *SSC v2* in the target zone, also passed transparently through it, and eventually reached *v2*.

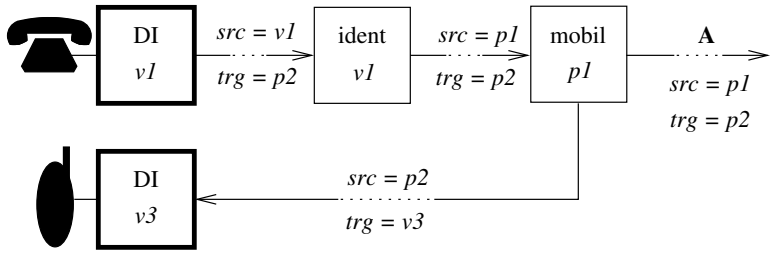

Figure 3: The mid-call move function of Personal Mobility.

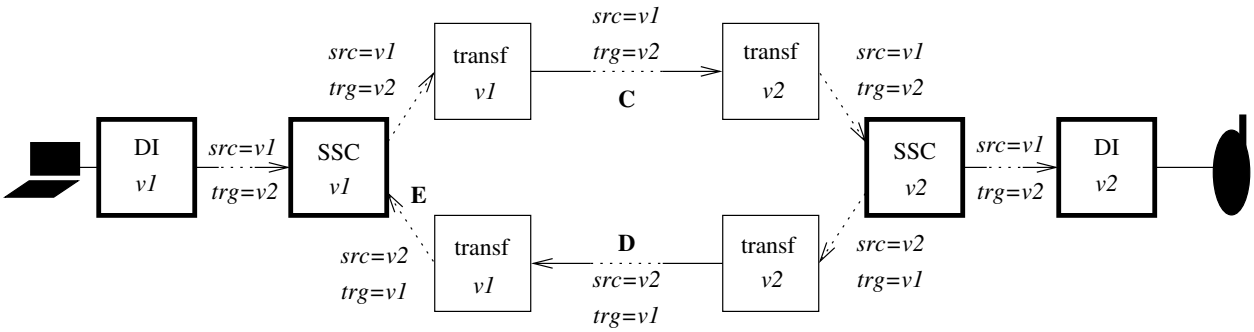

Figure 4: The Multiparty Control feature.

Later the user of *v2* instructed its *SSC* box to place a new personal call to *v1*.[1] *SSC v2* obeyed this command by reversing the setup signal of its outgoing internal call with *src = v1, trg = v2*, resulting in a setup signal with *src = v2, trg = v1*. This resulted in a chain of internal calls that ended at *SSC v1*.

SSC causes the signaling channel of the one internal call on its device side to be shared among the multiple internal calls on its network side. When *SSC v1* received the new incoming internal call from *v2*, it used this signaling channel to tell its user that a new personal call had arrived. At this point the user can enter a command to answer the new personal call and connect its voice channel to the voice channel of the device.

As a result of switching capabilities, an *SSC* box can maintain any number of personal calls, any of which can be outgoing or incoming from the perspective of its user. The user can connect the voice channel of his device to any one of them that he wishes; all the others are "on hold".

The *spontaneous conferencing* (as opposed to prescheduled conferencing) function of the *SSC* box enables its user to group personal calls into conferences any way he likes, and to change the groupings at any time. The members of the group can speak to each other at all times, and to the user whenever the user has switched his voice channel to that conference.

The *transfer* function enables a user to move his end of a personal call to another person (as opposed to moving it to another device he is using, which is the purpose of mid-call move). This function is performed by a *transf* box. The *transf* box is reversible, as it is equally meaningful to transfer both outgoing and incoming personal calls.

A *transf* box responds to a transfer command from its user. Figure 5 shows part of the usage that evolved from Figure 4 after the user of *v1* transfered the lower personal call to *v3*.

A user can also transfer an entire conference group. In this case all the personal calls in the conference receive transfer signals leading to the same destination.

The *SSC* and *transf* boxes of the Multiparty Control feature are separate for two reasons. Each one is perfectly useful without the other, so there is no necessity to subscribe to them both.

Equally important, most other feature boxes subscribed to by the same address go *between* the two boxes. In other words, *SSC* is ordered early in the source zone and late in the target zone, while *transf*

[1] This would not happen in real life, because the user of *v2* is already talking to the user of *v1*. It is convenient, however, to fit all the interesting cases into one diagram.

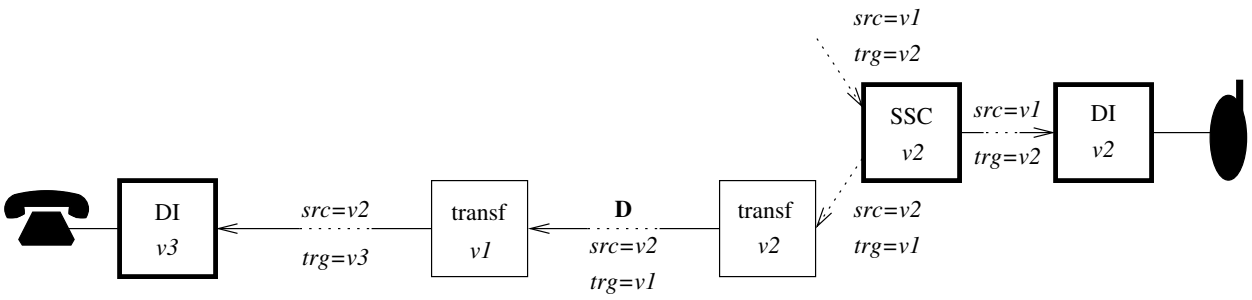

Figure 5: The transfer function of Multiparty Control.

is ordered late in the source zone and early in the target zone. In Figure 4, most of the target boxes of *v1* would be found in area **E**. The value of this arrangement is that when a user transfers his end of a personal call, most of his feature boxes are no longer included in it. In Figure 5, for example, area **E** is gone.

Note that quite a bit of UI signaling is needed for Multiparty Control. The user should know the status of all his personal calls, including their individual states and grouping into conferences. The user must be able to change the states of personal calls, regroup calls into conferences, switch among conferences, and transfer.

3.3 Device augmentation

Many telephones do not have the ability to display the status information generated by Multiparty Control, nor do they offer the user any way to generate the necessary commands. The *Device Augmentation* feature enhances the user interface of a telephone.

The *voice signaling* function enables the voice channel of a telephone to be used for signaling. When a feature box sends a signal containing status information, another feature box performing the voice signaling function intercepts the signal and translates it into a tone or announcement played to the user. When a user encodes a command in touch tones, a feature box performing the voice signaling function recognizes tone sequences and translates them into command signals. A box performing voice signaling can be located anywhere in the usage between the device interface and the feature box whose user interface it is supporting.

Even more powerful, a feature box can associate a telephone with a graphical user interface (GUI) running on a nearby computer. This is the primary function of the *compound device interface (CDI)* box shown in Figure 6. The box acts like an interface box for a device with two hardware components. The *CDI* box also includes a voice signaling function, to be used as a backup when the GUI is not present.

In the figure, *d1* and *d2* are both addresses of compound devices. *CDI* is reversible, and the figure shows *CDI d1* being routed to in the source zone, while *CDI d2* is being routed to in the target zone. Because *CDI* is a bound box, all personal calls to or from a compound device must go through the same *CDI* box. Like a true device interface, a *CDI* box can have only one internal call on its network side at a time.[2]

When *CDI d2* receives an internal call, it places corresponding internal calls to the address of its signaling device *s2* and the address of its voice device *v2*. If the call to *s2* fails then *CDI* provides a voice signaling function on the call to *v2*. If the call to *s2* succeeds then *CDI* uses its signaling channel for all user-interface signaling.

The personal call in Figure 6 was placed by the signaling device *s1* of compound device *d1*. Device *s1* subscribes in the source zone to the *assump* box, which performs the function of *assumption* of the identity of *d1* as source of the personal call. Because of assumption, the personal call is next routed through *CDI d1* in the source zone.

A compound device address is like a personal address in identifying a virtual entity within a telecommunication system. Just as a personal target address is located by a *mobil* box at a device address in Figure 2, a compound-device target address is located by a *CDI* box at two device addresses in Figure 6. Given this similarity, it should come as no surprise that the relationship of *ident v1* and *mobil p1* on the source side of Figure 2 is exactly the same as the relationship between *assump s1* and *CDI d1* on the source side of Figure 6. *Ident* and *assump* are just different names for the same feature box.

[2]DFC imposes this restriction on device interfaces so that features, for example Multiparty Control, are always modeled as being provided by feature boxes rather than built into devices.

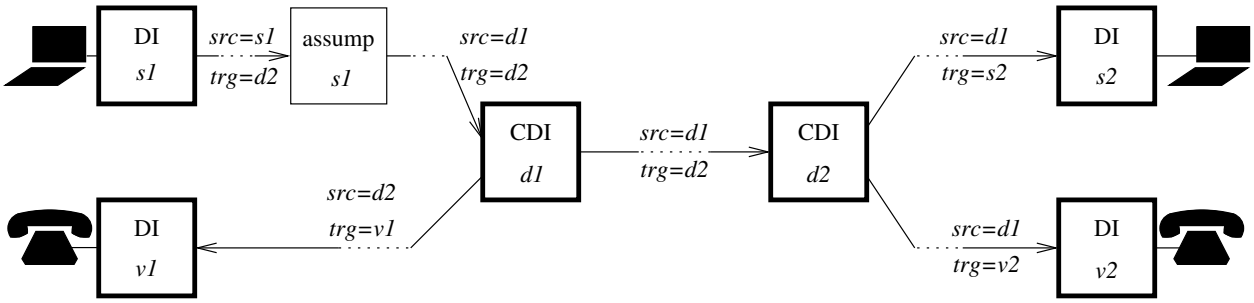

Figure 6: The Device Augmentation feature.

When *CDI d1* receives its incoming call, it places two outgoing calls. First it applies the *continue* method to the incoming call, and uses the result to place an outgoing call to *d2*. Next it applies the *reverse* method to the outgoing call, and uses the result to place an outgoing call with *src = d2, trg = v1*. The telephone with address *v1* is the voice device of *d1*.

3.4 Other features

We have presented Personal Mobility, Multiparty Control, and Device Augmentation in some detail because they are complex features with many functions and interactions. They are the ones requiring the most detailed analysis. All have boxes that are bound, reversible, or both, which is typical of the most challenging features.

Fortunately, such features are usually in the minority. Our service also includes a Voice Mail feature and a Call Log feature, both of which interact much less with other features. These will be discussed briefly in Section 5.3.

4 Configurations

Depending on personal needs, device characteristics, and network capabilities, the three features can be combined in many different ways. This section describes some of the most useful configurations.

The most important point about these configurations is that they are not achieved by programming. To get a configuration, all one needs to do is subscribe the right addresses to the right feature box types, and give the right address data to features that translate from one address to another. Then, as personal calls are processed, the DFC routing algorithm will automatically assemble feature boxes into the correct usages.

The assembly is dynamic, and is often influenced by the behavior of feature boxes in changing addresses. Thus it is truly a mechanism for component coordination, and not just a code-configuration mechanism.

Because these configurations are so heavily influenced by addresses, and always have several addresses, it is important to keep them straight. We use symbolic addresses in five different categories; the symbols are assigned to categories as follows:

an address in category represents:	*addresses:*
person	*p1, p2*
personal voice channel	*c1, c2*
compound device	*d1, d2*
voice device	*v1, v2, v3*
signaling-only device	*s1, s2*

First we consider configurations without Device Augmentation. Personal Mobility and Multiparty Control compose in two ways. If some devices have UI signaling capabilities good enough for Multiparty Control and some do not, it is best for only the addresses of devices with good signaling to subscribe to Multiparty Control. Let us assume that *v1* and *v2* are two such devices. Then all of Figure 2 except its interface boxes and devices would be found nested within area **C** of Figure 4. (For the addresses to match, the original target address of the upper personal call in Figure 4 would have to be *p2*, and the ultimate source address would have to be *p1*). The same subgraph, with *v1/v2* reversed and *p1/p2* also reversed, would be found nested within area **D**. In this configuration, a user can only move one personal call at a time with the mid-call move function.

If all or most devices can support the signaling needed for Multiparty Control, on the other hand, it makes sense for personal addresses to subscribe to Multiparty Control. In this case all of Figure 4 except its interface boxes and devices, with *p1* substituted for *v1* and *p2* substituted for *v2*, might be found nested within the **A/B** area of Figure 2. In

this configuration, the mid-call move function moves all of a user's personal calls simultaneously.

Next we consider configurations with Device Augmentation. Figure 7 is just the composition of Figures 2 and 6, so persons *p1* and *p2* are currently located at compound devices *d1* and *d2*, respectively.

In this configuration, the compound devices could subscribe to Multiparty Control. If so, the *SSC* and *transf* boxes of *d1* would be located at **F** along the path of the personal call shown in Figure 7, and the *SSC* and *transf* boxes of *d2* would be found at **G**. Alternatively, the personal addresses could subscribe to Multiparty Control. In that case the *SSC* and *transf* boxes of *p1* and *p2* would be found at **H** and **J**. Either way, Device Augmentation is placed where it needs to be, between the actual devices and the Multiparty Control feature boxes that do so much signaling.

In Figure 7, other source-zone feature boxes of *p1* would be found in area **H**. These could include boxes for personalized dialing, retrieving voice mail, and call logging. Other target-zone features of *p2* would be found in area **J**. These could include boxes for selective handling of personal calls, recording voice mail, and call logging.

In understanding Figures 6 and 7, you are invited to think of a computer and a telephone sitting next to each other on a desktop. That is not, however, the only possible use of Device Augmentation. Either voice address *v1* or *v2* might identify a cellphone with its own network-based device mobility. Either signaling address *s1* or *s2* might identify an Instant-Messenger-like GUI with its own network-based device mobility. These examples show that the pairing between voice and signaling devices need not be fixed!

Figure 8 combines Personal Mobility and Device Augmentation in a completely different and very useful way. Here the personal addresses subscribe to Device Augmentation. Each person has a signaling device address (*s1* or *s2*) that identifies a GUI with device mobility. A person can log into a copy of the GUI on any computer, and use it from any network connection. Because of the use of network-based device mobility, there is no need for software-based feature mobility on the signaling side.

On the voice side, a personal *CDI* box uses a personal voice channel address *c1* or *c2*. Because a person might want to locate his personal voice channel at any telephone, addresses *c1* and *c2* subscribe to Personal Mobility.

Both Figures 7 and 8 depict personal calls from *p1* to *p2*. In Figure 7 the user initiated the personal call from his GUI, but in Figure 8 the user initiated the personal call from his telephone. The figure shows how, on the source side, Personal Mobility applies to *c1*, and *c1* assumes its role as the voice channel of the personal compound device of *p1*.

In the configuration of Figure 8, Multiparty Control can only be subscribed to by the personal addresses, and its feature boxes can only be found at **K** and **L**. This is the only way that Multiparty Control can take advantage of Device Augmentation. Other source-zone feature boxes of *p1* would also be found in area **K**; other target-zone features of *p2* would also be found in area **L**.

5 Management of feature interactions

A feature interaction is some way in which a feature modifies or influences another feature in defining overall system behavior. Feature interaction is an inevitable by-product of modularity in a feature-oriented description [27].

A particular feature interaction can be positive (desirable) or negative (undesirable). To manage feature interactions successfully, it is necessary to understand all the potential interactions, to distinguish which are good and which are bad, to enable all the good ones, and to prevent all the bad ones. Our discussion of this management process is divided according to the major causes of feature interaction.

5.1 Feature interactions caused by address translation

Clearly feature boxes interact with other feature boxes through their choices of when to place internal calls and what addresses to give them. These choices affect who a user is talking to, what user identification a person or feature box sees, what address is available to return a personal call, and which features are invoked by appearing in the usage. Although these feature interactions are quite complex, we have made a start at understanding them [25].

One of the most difficult aspects of managing feature interactions is deciding which of several possible interaction behaviors is the most desirable. To help determine how features *should* interact, we propose some principles for global behavior of telecommunication systems. These principles should be respected, regardless of how many features there are or how complex their behavior. For example (paraphrased loosely from [25]):

- *Privacy:* Use of a more public address such as a personal address conceals from the far party more private addresses such as device addresses.

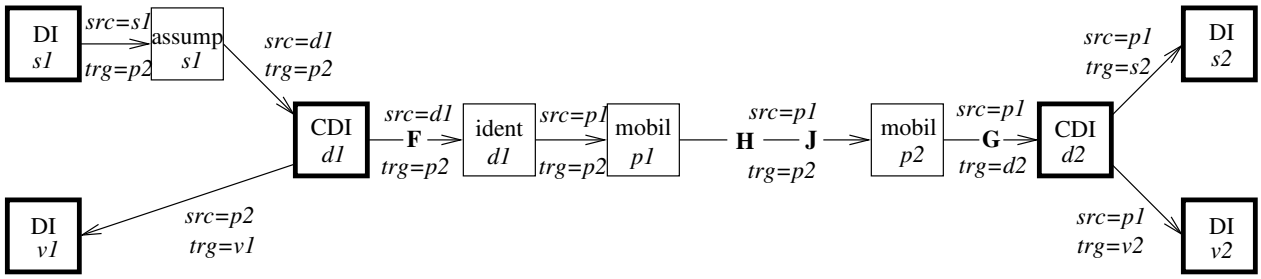

Figure 7: A composition of Personal Mobility and Device Augmentation.

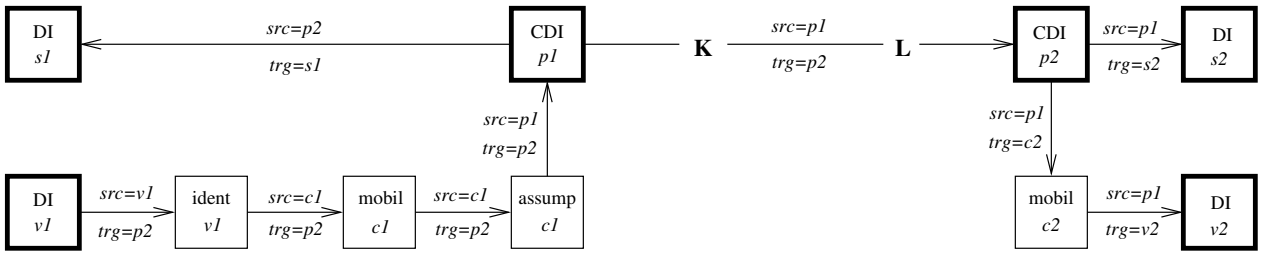

Figure 8: Another composition of Personal Mobility and Device Augmentation.

- *Authenticity:* If an address subscribes to an authentication feature, then it cannot be used as caller identification without the participation of the address's owner.
- *Returnability:* A target feature or callee can return a call to the entity that initiated the call by targeting the source address it has received.

Within the context of an architecture such as DFC, it is possible to define formal properties that correspond to these principles, and to formulate constraints on the behavior of features. It can then be proved that if features obey the constraints, global system behavior satisfies the properties [25].[3]

The constraints are based on the notion of partitioning addresses into *address categories* according to what they identify. There is a partial order on address categories called *abstraction*. By definition, a personal address is more abstract than the address of a device where the person is located, a group address is more abstract than the personal address of a member of the group, and a public address is more abstract than a private one. The most important constraints are:

- An application of the *continue* method in the source region only translates a source address to a more abstract source address, and an application of the *continue* method in the target region only translates a target address to an equally concrete or more concrete target address.
- An application of the *continue* method in the target region does not change the source address.

For our feature set, the address categories are those listed in the table in Section 4. They are listed in order from most abstract to most concrete, except that the categories of voice device and signaling-only device are unordered with respect to each other.

With the assumptions that there are no addresses outside the five categories, and that users only dial personal addresses, the configurations in Section 4 adhere to the constraints. Our service allows any set of subscriptions and address data, provided that the address translations adhere to the constraints, which means that many other configurations are possible. Some will be uncommon, and some will be downright surprising.

For all these configurations, however unanticipated, the global behavior of the service has the properties of privacy, authenticity, and returnability. Equally important, usages have a *monotonicity* property: as a source region grows, the addresses of the feature boxes become successively more abstract; as a target region grows, the addresses of the feature boxes become successively more concrete. This additional structure helps us to manage other feature

[3]The results in [25] apply to a broader range of telecommunication sytems than just those built with DFC. For example, they solve problems with electronic mail as identified by Hall [11].

interactions, for example those related to shared signals (Section 5.3).

In real use, the presence of external addresses (addresses not in any of these categories) will weaken the results somewhat. The results remain valid, however, as an analysis of how the features of the service interact with each other.

There is a major limitation in the results of [25], which is that they only apply to chains of internal calls all placed in the same direction. In other words, they apply to the *continue* method and not the *reverse* method.

We are currently engaged in removing this limitation, and extending our understanding of address translation to usages in which a path between two endpoints can consist of linked subchains set up in alternating directions. This is very challenging, but we are making steady progress toward the goal [26]. Some of the issues are described in Section 7.1.

5.2 Functional dependence

Ideally each feature would offer an independent new telecommunication capability to its users. The real world is usually messier. An increment of functionality designated as a feature might provide infrastructure to be used by other features. In such a case the other features are functionally dependent on the infrastructure feature; this functional dependence is a beneficial feature interaction that must be supported.

Device Augmentation is an infrastructure feature. It improves the user interface for other complex features such as Multiparty Control.

For the most part, its interactions have already been managed. For example, the explanation of Figure 7 pointed out that Multiparty Control could be subscribed to by any of *d1, d2, p1,* or *p2*, placing its feature boxes at **E, F, G,** or **H**, respectively. In all those cases there is a *CDI* box between the Multiparty Control feature boxes and their user, providing the enhanced signaling desired.

Figure 8, on the other hand, has an unresolved problem. The *mobil* boxes of *c1* and *c2* are outside the areas of the usage (namely **K** and **L**) where they could benefit from the user interface provided by *CDI*. They cannot be moved there because it is the essence of this configuration that Personal Mobility applies to the voice channel only.

To solve this problem, we used a different version of the Personal Mobility feature in which its functions are decomposed into boxes in a different way. Now the *mobil* box performs only the core functions of location and mid-call move. The new *mobil UI* box performs authentication, update, and the UI portion of mid-call move (when *mobile UI* receives a move command, it sends a private move signal toward the *mobil* box).

Figure 9 shows the benefits of this decomposition. The personal voice channel addresses *c1* and *c2* subscribe to *mobil*, so that mobility applies to the voice channel only. The personal addresses *p1* and *p2* subscribe to *mobil UI*, so that the user interface of Personal Mobility can take advantage of Device Augmentation. With the one exception of a private move signal, the boxes of Personal Mobility communicate only through operational data.

5.3 Feature interactions caused by shared signals

Sections 4 and 5.1 focused on the routing algorithm, which is one of the two major mechanisms for component coordination in DFC. The other major mechanism for component coordination is signaling. Signals travel on the signaling channels of internal calls, and are propagated by boxes from one internal call to another, according to the DFC protocol and conventions on the meanings of shared signals.

All distributed applications have components that exchange signals. Often, however, a component plays a unique role or corresponds to an essential resource. DFC is somewhat unusual in adapting to an interactive context the pipes-and-filters idea that each component is optional. The DFC protocol has been carefully designed to ensure that there is never an externally observable difference between a single internal call and a chain consisting of multiple internal calls and feature boxes.

One result is that DFC signaling is very successful at avoiding unnecessary interference (bad interactions) among features. For example, a particular personal call might encounter Multiparty Control or Device Augmentation features at *both* the personal and device levels. These configurations are not recommended because users are likely to find their behavior confusing. Nevertheless, they will function without failing. Their behavior will even be predictable to someone who understands the full configuration.

Feature interactions caused by shared signals are not yet understood as thoroughly as feature interactions caused by address translation. Nevertheless, there is an initial version of an algorithm for detecting potential interactions of this kind [24].

For managing these interactions, we rely so far on signaling conventions within the DFC protocol. The most important convention concerns the use of the shared outcome signals.

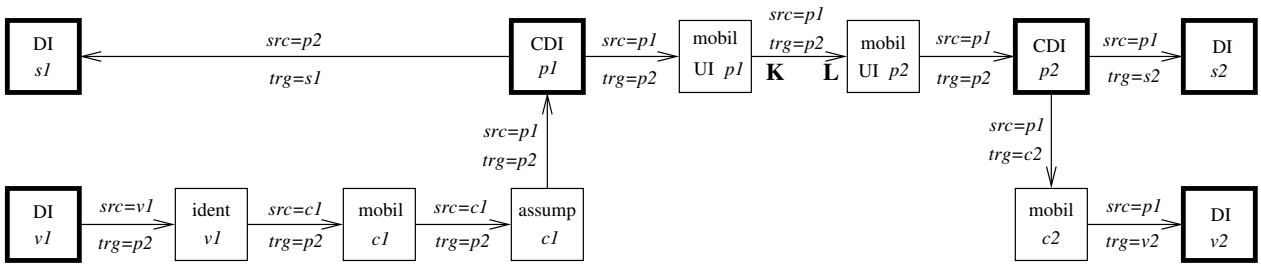

Figure 9: A better version of Figure 8.

The outcome signals *avail* and *unavail* indicate success and failure, respectively, of a personal call.[4] For the features in this case study, each internal call should transmit at most one of these signals. If a feature box receives an outcome from downstream it should send an outcome upstream, but only after completing any active phase triggered by receipt of the outcome. These conventions ensure that an outcome signal is an unambiguous token that travels upstream through the feature boxes of a personal call; if feature boxes interpret it as permission to be active, then their active phases are sequentialized by it. The precedence order on feature boxes conveys priority to respond to outcomes, with the boxes further downstream having higher priority.

For example, consider the personal target zone shown partially in Figure 10. Between the familiar *transf* and *SSC* boxes for Multiparty Control, there is a *call log* box and a *voice mail (VM)* box for recording voice mail if the features' subscriber cannot be reached.

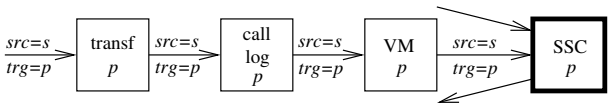

Figure 10: Personal target-zone features coordinated through conventions on the use of outcome signals.

Figure 10 shows an incoming personal call reaching an *SSC* box that is already engaged with other personal calls. If the callee signals *SSC* to answer the new personal call, then *SSC* will send *avail* upstream on this personal call and open the voice channel to the caller. If there is a timeout before the callee answers, then *SSC* will send *unavail* instead.

The *avail* signal makes *VM* transparent, while the *unavail* signal triggers it to open a voice channel to the caller and offer voice mail. Recording does not end until the caller disconnects and the resulting chain of teardowns propagates all the way to *VM*. *VM* will never propagate the *unavail* upstream, simply because the usage is torn down before it can do so.

The *call log* feature box puts a call record in a database. Note that both the user's answering a call and the triggering of voice mail cause a voice channel to the caller to be opened, although for very different purposes. The *call log* box can distinguish these cases by the presence or absence of an *avail* signal.

The coordination of *VM* and *transf* is more subtle because *transf* is triggered by a transfer command rather than an outcome signal. The signals are truly independent. For example, a callee can issue a transfer command before or after an *avail* outcome. In other words, the callee can transfer a personal call before or after answering it.

As a result, there can be a race condition: a transfer command can be followed closely on the upstream signaling path by a timeout-generated *unavail*. This will cause *transf* and *VM* to be triggered at about the same time.

Both *transf* and *VM* function by tearing down their downstream internal calls, which are irrelevant to their functions, and manipulating their upstream connections to the caller. This pattern of behavior inevitably conveys priority to the upstream feature box, because it is closer to the caller and can cut the downstream feature box off from the caller. We achieve the desired behavior, which is giving priority to the transfer function, by placing *transf* upstream of *VM*.

As this example shows, the signaling interactions among features are determined by their handling of signals and their position in the usage. Among feature boxes in the target zone of an address, for example the feature boxes in Figure 10, relative position is determined by the *precedence* order on feature box types.

[4]For simplicity, the third outcome signal *unknown* is omitted in this discussion.

Between feature boxes in different zones of a target region, on the other hand, precedence is irrelevant; relative position of the zones is determined by address translation. This is why monotonicity helps manage signaling interactions, as mentioned in Section 5.1. In a usage with monotonicity, the zones are ordered according to the abstraction level of their subscribing addresses, and this ordering can be used to ensure correct interactions [25].

6 Implementation

All the telecommunication features mentioned in this paper are running on our IP-based implementation of DFC [2]. We use multiple configurations to meet the needs of different groups of trial users.

We had the first versions of these features running about two months after the inception of the project. To meet this schedule we made use of many previously implemented feature boxes, either reusing them directly or modifying them for present purposes.

One design dimension that has been completely missing from our discussions and diagrams so far is the *network node* at which a process is running. It is worth noting that a DFC internal call can have an inter- or intra-node realization. All the feature boxes appearing in usages on behalf of the same address are implemented at the same network node, so the internal calls that connect them are intra-node calls. If the feature boxes of two addresses are implemented on different network nodes, on the other hand, an internal call connecting a box of each address is an inter-node call. Thus implementations of the usages discussed in this paper might be centralized or distributed.

This paper is concerned with the logical and functional implications of personal mobility. Many other issues, particularly efficiency issues, arise at the implementation level.

Most prominently, within the DFC architecture the signaling path of a personal call always passes through the features of the targeted person, regardless of where the person is located. If the personal call is ultimately routed to a device on a different node than the node hosting the person's features, there will be a permanent "hairpin" in the signaling path of the personal call. Such hairpins are not popular, and there is much interest in protocols for various mobile IP applications that remove the hairpins as soon as connections are set up.

We take a different view. If there are functions of personal features that operate during a call, rather than merely at its setup time, then the signaling path *must* pass through the personal features throughout the call. Of the feature boxes in this paper, *mobil, SSC, transf, CDI, mobil UI, call log,* and *VM* can be subscribed to by personal addresses, and have mid-call functions. This is abundant evidence that, for the telecommunications application at least, mid-call features are the rule rather than the exception.

Separation of signaling and media has been a goal of telecommunication architectures for a long time. We use this well-established principle to solve the real problem, which is efficient transmission of media. Our IP-based implementation of DFC [2] optimizes bandwidth by transmitting media along a more direct route than the signaling path [8].

7 Conclusions

7.1 Lessons learned

When this project began there was no *reverse* method in DFC. As we designed the features to fit within the DFC architecture, we were troubled by a long list of awkwardnesses and anomalies. Finally we faced the problem, and discovered the simple truth illustrated by Figure 11.

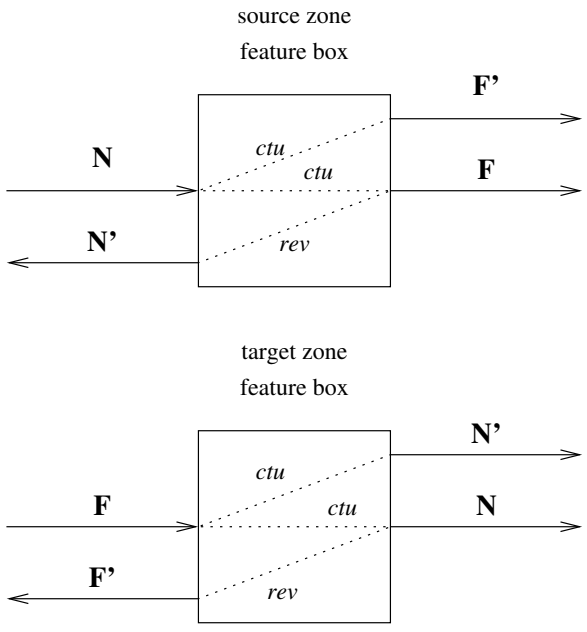

Figure 11: Motivation for the *reverse* method.

At the top of the figure is a feature box of type t routed to in the source zone of address a. Its incoming call **N** is the means by which it is connected to its own subscriber, or the *near party*. Behaving typically, it first applied *continue* to the setup of **N**, and

used the resulting setup signal to place an outgoing call **F**. **F** is the means by which this box is connected to a *far party*.

Sometime during its lifetime, the box may have to re-establish, replace, or augment one of these connections. To replace **F** by **F'**, it can simply do another *continue*. The only correct way to replace **N**, on the other hand, is to apply *reverse* to the setup signal of **F**. In placing **N'**, the box is now behaving like a target-zone feature box. The setup signal has *region* = *target*, and it should be routed to the next box type after t in the target zone of a.

As the bottom of Figure 11 shows, the situation of a target-zone feature box is symmetric. Its original incoming call **F** comes from the far party, and its original outgoing call **N** goes to the near party. To replace **N** by **N'**, it can simply do another *continue*. The only correct way to replace **F**, on the other hand, is to apply *reverse* to the setup signal of **N**, and to place a call as if it were a source-zone feature box.

For reverse routing to work correctly, the *precedence* partial order must have the following properties:

- Reversible box types are totally ordered.
- The order of reversible box types in the source zone is the reverse of the order of reversible box types in the target zone.

With these properties, it is possible to prove that a "zone" of adjacent boxes with the same address has a fixed sequence of reversible box types (traversing from the far-party end to the near-party end), even if it was formed from sub-zones of the address, set up in alternating directions.

In this feature set, the reversible box types are *mobil, SSC, transf,* and *CDI*. Although the problems solved by *reverse* had been observed occasionally in the past, no general conclusions had been drawn from them. It took an onslaught of similar observations to get our attention. The result was the first major change to DFC in three years.

One lesson that is continually being reinforced, by this project and by others we have undertaken, is the dominance of user-interface issues. People expect VoIP technology to provide them with advanced, enhanced features. Advanced, enhanced features require advanced, enhanced user interfaces so that people can control them. Yet the user-interface technology is often the oldest, least flexible part of the service. As a result, a large proportion of the effort involved in developing every VoIP service is expended on the user interface.

The other major surprise in carrying out this work is the extent to which personal mobility is about configuration. This arises from the fact that features can be associated with devices, persons, or both. Different associations (subscriptions) lead to radically different capabilities and feature interactions. Different customers wanted different combinations of these. This made it necessary to find a way to understand and manage many configurations without expending a great deal of separate effort on each one.

7.2 Related work

7.2.1 SIP-based VoIP

There are now numerous efforts, both academic and commercial, to developed VoIP telecommunications services. Many of these are using SIP [19, 20], which has emerged as the dominant protocol.

Some studies of feature interaction in VoIP services consider the same features that are familiar from the PSTN, for example Caller Identification, Call Forwarding on Busy, Call Waiting, and Three-Way Calling [6, 23]. Rather than analyzing existing features, we are concerned with designing new features as well as possible. We consider Caller Identification to be a birthright of every call, not an optional feature. Call Forwarding on Busy is subsumed by our more general Personal Mobility. Call Waiting and Three-Way Calling are subsumed by our more general Multiparty Control.

In SIP terminology, an *application server* is a platform on which features are provided. Work on commercial SIP application servers, at least as disclosed publicly, has not gone much beyond basic building blocks such as detecting DTMF tones [15, 18]. Such capabilities have long been present in our platform, which is itself usable as a SIP application server [2].

7.2.2 ICEBERG and the Mobile People Architecture

The ICEBERG project [17, 22] and the Mobile People Architecture (MPA) [16] are like our case study in focusing on how personal mobility *should* look in VoIP. The three projects overlap and complement one another.

Considering the functions of personal mobility, both ICEBERG and MPA explore in detail the semantics of the location function: how does a system locate a person? There are many questions to answer about what data is used, how it is obtained, where and how it is maintained, how it is combined to make decisions, etc. In contrast, our location function answers these questions in the most rudimentary way.

ICEBERG offers identification and mid-call move functions, but is not much concerned with user-interface issues. MPA deals with the difference be-

tween devices by offering various media and protocol conversions, but does not provide identification or mid-call move. A goal of our work is to consider all functions related to personal mobility, even though some functions are provided in their simplest forms.

Although both ICEBERG and MPA aim to offer personal mobility in an extensible context, so that it can be combined with other telecommunication features, they are vague on how this will work. This is worrisome because the long history of the feature-interaction problem in telecommunications indicates that few software challenges are bigger than extending a telecommunication service with new features.

In contrast, the greatest strength of our work is that it is based on a formally defined architecture in which adding features and managing feature interactions are well-understood. Even though there are still many unanswered questions about these activities, there is also a clearly marked path toward answering them.

7.2.3 Agent systems

There is a wide range of distributed applications. Some use signaling for its own sake (electronic commerce) while some use it to set up connections for large-scale transfers of data. Some data connections have real-time constraints because they are actually media transmissions, while others do not. Some applications connect a single user to network services (broadcast, Web services, content-on-demand), while others connect users together.

Telecommunications is a distributed application that emphasizes real-time communication among people. Nevertheless, its functions overlap with the functions of related distributed applications such as broadcast and Web services. In fact, other feature bundles developed on our infrastructure include telecommunication services integrated with Web services. And all distributed applications share common concerns rooted in the exigencies of networking. Thus it seems likely that the component-coordination mechanisms used in DFC would prove useful to other distributed applications besides telecommunications.

This conjecture seems supported by Griss [10], who says that *agent systems* have great potential in electronic commerce. Griss defines an agent system as a component system with some of the characteristics of adaptability, autonomy, collaboration, knowledgeability, mobility, and persistence.

Griss describes agent systems as employing varying levels of "choreography" among agents. Sometimes agent communication is semantically rich and loosely constrained, while at other times or between other agents it is semantically rigid and tightly constrained—as telecommunication protocols are.

DFC fits in one end of this spectrum, and its component-coordination mechanisms have proven notably successful in achieving autonomy and collaboration as well as component reuse. For these reasons, the same mechanisms may be useful in other agent architectures.

7.3 Evaluation

The VoIP service presented in this paper is the second major project that has built, on a DFC platform, features to meet customer requirements. Although this project brought some surprises that caused some changes to DFC, the surprises have not weakened our confidence in DFC. Rather, they revealed to us some symmetries in the telecommunication domain that were not previously apparent. The new version of DFC is both simpler and more general than the old one.

Subsequent organizational changes have prevented this feature set from getting much use, so there has been no opportunity to judge or polish its detailed behavior. Hence the primary contribution of this work is architectural. It elucidates the functions of personal mobility and several other major features, and shows how these functions compose and interact. Within this framework, complex services can be developed with both quality and efficiency.

Acknowledgments

Greg Bond, Eric Cheung, Karrie Hanson, Don Henderson, Gerald Karam, Hal Purdy, and Ken Rehor made many contributions to this work, including building and maintaining the infrastructure on which these features run. Andrew Forrest improved the presentation.

References

[1] D. Amyot and L. Logrippo, editors. *Feature Interactions in Telecommunications and Software Systems VII*, IOS Press, Amsterdam, 2003.

[2] Gregory W. Bond, Eric Cheung, K. Hal Purdy, Pamela Zave, and J. Christopher Ramming. An open architecture for next-generation telecommunication service. *ACM Transactions on Internet Technology* IV(1), February 2004, to appear.

[3] L. G. Bouma and H. Velthuijsen, editors. *Feature Interactions in Telecommunications Systems*, IOS Press, Amsterdam, 1994.

[4] M. Calder and E. Magill, editors, *Feature Interactions in Telecommunications and Software Systems VI*, IOS Press, Amsterdam, 2000.

[5] E. Jane Cameron, Nancy D. Griffeth, Yow-Jian Lin, Margaret E. Nilson, William K. Schnure, and Hugo Velthuijsen. A feature-interaction benchmark for IN and beyond. In [3], pages 1-23.

[6] Ken Y. Chan and Gregor v. Bochmann. Methods for designing SIP services in SDL with fewer feature interactions. In [1], pages 59-76.

[7] K. E. Cheng and T. Ohta, editors, *Feature Interactions in Telecommunications III*, IOS Press, Amsterdam, 1995.

[8] Eric Cheung, Michael Jackson, and Pamela Zave. Distributed media control for multimedia communications services. In *Proceedings of the 2002 IEEE International Conference on Communications: Symposium on Multimedia and VoIP—Services and Technologies*, IEEE Communications Society, 2002.

[9] P. Dini, R. Boutaba, and L. Logrippo, editors. *Feature Interactions in Telecommunication Networks IV*, IOS Press, Amsterdam, 1997.

[10] Martin L. Griss. Software agents as next generation software components. In G. T. Heineman and W. T. Councill, *Component-Based Software Engineering,* pages 641-657. Addison-Wesley, 2001.

[11] Robert J. Hall. Feature interactions in electronic mail. In [4], pages 67-82.

[12] Michael Jackson and Pamela Zave. Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering* XXIV(10):831-847, October 1998.

[13] Michael Jackson and Pamela Zave. The DFC Manual. AT&T Research Technical Report, February 2003. Available at `http://www.research.att.com/projects/dfc`.

[14] K. Kimbler and L. G. Bouma, editors. *Feature Interactions in Telecommunications and Software Systems V*, IOS Press, Amsterdam, 1998.

[15] Anders Kristensen. SIP Servlet API, Version 1.0. Dynamicsoft, Inc.

[16] Petros Maniatis, Mema Roussopoulos, Ed Swierk, Kevin Lai, Guido Appenzeller, Xinhua Zhao, and Mary Baker. The Mobile People architecture. *ACM Mobile Computing and Communications Review* I(2), July 1999.

[17] Bhaskaran Raman, Helen J. Wang, Jimmy Shih, Anthony D. Joseph, and Randy H. Katz. The ICEBERG project: Defining the IP and telecom intersection. *IEEE IT Professional,* November/December 1999, pages 38-45.

[18] Jonathan Rosenberg, Peter Mataga, and Henning Schulzrinne. An application server component architecture for SIP. IETF Internet Draft, SIP Working Group, 2 March 2001.

[19] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. IETF Network Working Group, Request for Comments 3261, 2002.

[20] Jonathan D. Rosenberg and Richard Shockey. The Session Initiation Protocol (SIP): A key component for Internet telephony. *Computer Telephony* VIII(6):124-139, June 2000.

[21] Mary Shaw and David Garlan. *Software Architecture.* Prentice-Hall, 1996.

[22] Helen J. Wang, Bhaskaran Raman, Chen-nee Chuah, Rahul Biswas, Ramakrishna Gummadi, Barbara Hohlt, Xia Hong, Emre Kiciman, Zhuoqing Mao, Jimmy S. Shih, Lakshminarayanan Subramanian, Ben Y. Zhao, Anthony D. Joseph, and Randy H. Katz. ICEBERG: An Internet core network architecture for integrated communications. *IEEE Personal Communications,* August 2000, pages 10-19.

[23] T. Yoneda, S. Kawauchi, J. Yoshida, and T. Ohta. Formal approaches for detecting feature interactions, their experimental results, and application to VoIP. In [1], pages 205-212.

[24] Pamela Zave. An experiment in feature engineering. In A. McIver and C. Morgan, editors, *Programming Methodology*, pages 353-377. Springer-Verlag, 2003.

[25] Pamela Zave. Ideal address translation: Principles, properties, and applications. In [1], pages 257-274.

[26] Pamela Zave. Ideal connection paths. AT&T Research Technical Report, in preparation.

[27] Pamela Zave. Requirements for evolving systems: A telecommunications perspective. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, pages 2-9. IEEE Computer Society, 2001.