Mid-Call, Multi-Party, and Multi-Device Telecommunication Features and Their Interactions

Pamela Zave
AT&T Laboratories—Research
Florham Park, New Jersey, USA
pamela@research.att.com

ABSTRACT

Mid-call telecommunication features act only after an initial media connection has been set up. Many mid-call features perform complex functions such as handling multiple far parties and multiple devices. They cause many important feature interactions that are little known and poorly understood. Using the example of personal features associated with a mobile phone, this paper presents a general method for analyzing the interactions of mid-call features with each other and with other features in a service. The paper discusses how to determine the most desirable behavior of interacting features. It also shows, as a proof of existence, how to manage these interactions in pipes-and-filter implementations of telecommunication services.

1. INTRODUCTION

Interaction of telecommunication features is a problem with a long history and large literature. Although the problem was first recognized and addressed in circuit-switched telephone systems, it arises relatively unchanged in IP telecommunications.

Many features are invoked when voice or multimedia calls are being set up. Call set-up features and their interactions are fairly well-understood by now.

Mid-call features act only after an initial media connection has been set up. Many mid-call features perform complex functions such as handling multiple far parties and multiple devices. Although much has been written about call waiting and conferencing in particular, there are still many important interactions caused by mid-call features that are little known and poorly understood. The purpose of this paper is to explain how to analyze these interactions among the features of a service, how to determine the most desirable behavior of interacting features, and how the interactions might be managed in practical implementations.

Bruns argues that there are many possible definitions of "feature interaction," depending primarily on the answers to two questions [2]:

- Are there independent specifications of feature behavior?
- In the implementation, is there a capability for implementing features separately and composing those implementations?

For purposes of this paper the answers are that there must be a feature capability in the implementation, but not necessarily a specification of each feature. For these answers one of the applicable definitions of feature interaction is that two features interact if their composition is not well-formed. This is the definition used in this paper, where "not well-formed" might mean that the composition is inconsistent or erroneous in some way, or might mean that the definition of the composition is incomplete. A composition would be incomplete if it raises new questions that must be answered before the implementation can be finished. The feature interactions in this paper cause a few inconsistencies and raise many new questions.

Section 2 presents the network context and feature capability assumed in this paper. The context is that features are implemented in application servers in the network rather than in devices or endpoints. Network features can be significantly different from device or endpoint features, and thus have different interactions. For example, call waiting implemented in an endpoint requires multiple signaling channels between the network and endpoint [11], while call waiting implemented in the network requires the use of only one signaling channel between network and device.

The feature capability is pipes-and-filters composition, as introduced in the telecommunications domain by the Distributed Feature Composition (DFC) architecture [7, 17]. Basic pipes-and-filters composition is now also available in the SIP Servlet standard [8] and IMS [1], and is reviewed in Section 2.

Sections 3 through 6 present a comprehensive analysis of mid-call feature interactions. The analysis is organized according to various feature categories. All of the pieces are brought together and summarized in Section 7.

Sections 3 through 6 not only expose interactions, but explain how they can be managed successfully. The first step in managing a feature interaction is to decide how the features *should* interact. In terms of the definition of a feature interaction as an ill-formed composition, it is necessary to decide how to resolve an inconsistency, or to answer an unanswered question.

The behavior of a telecommunication system should not be designed in a vacuum, but rather by thinking about how people will use it. To make such thinking concrete, this paper focuses on personal features. More specifically, it con-

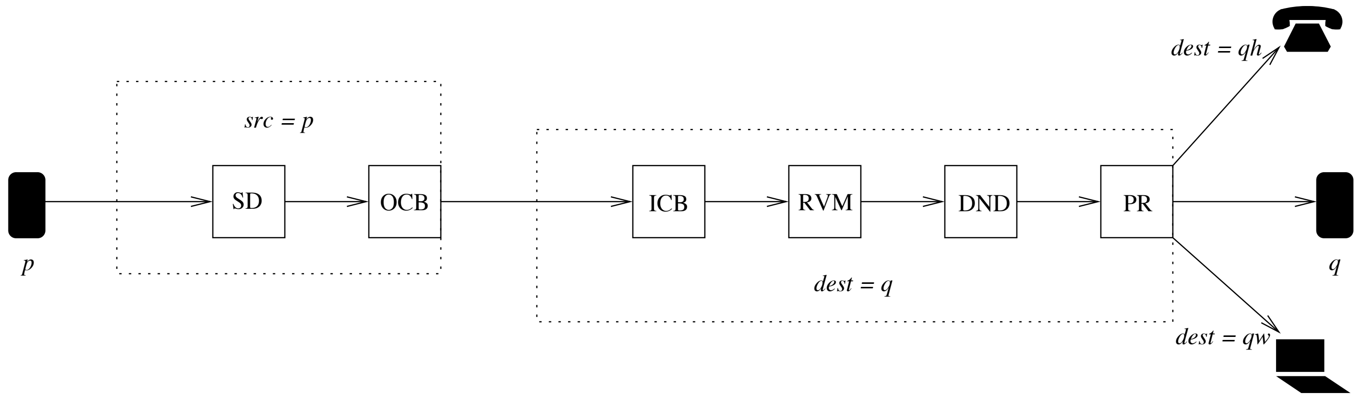

Figure 1: A usage formed to implement one call. Each solid box denotes a call set-up feature. Each arrow denotes a dialog, set up in the direction of the arrow.

cerns the features associated with a telephone number that is expected to be used by a single individual. Most commonly this would be a mobile-phone number, because home or work telephones are often shared, and people carry their mobile phones with them everywhere. The word *subscriber* is used in this paper to refer to both the person and the telephone number.

The decision to focus on personal features is compatible with the decision to focus on network implementations. For instance, call waiting for mobile phones is implemented in the network, because a network implementation conserves the scarce resource of wireless channels.

The second step in managing a feature interaction is to adjust the feature implementations or composition so that features interact as they should. In Sections 3 through 6 this is accomplished by using capabilities of DFC beyond the basic ones reviewed in Section 2. Section 8 completes the presentation by explaining how these capabilities can be implemented in SIP Servlets and/or IMS.

The feature-interaction management in this paper is only an existence proof, in the sense that it is possible to implement the correct behavior in other ways than using the advanced capabilities of DFC. The feature interactions themselves are real and independent of DFC, however, so that they must be managed somehow in any successful implementation of mid-call features.

Historically many reported feature interactions were created by user interfaces too simple to control complex features well. For example, there is a well-known user-interface feature interaction between call waiting and conferencing on plain touch-tone phones [3]. With the rich user interfaces of today's mobile phones, this is no longer a problem, and user interfaces are largely ignored here.

2. FEATURE COMPOSITION

2.1 Usages and features

A *call* is an attempt by one communicating entity (usually, but not always, a person) to connect to another communicating entity via telephone. Figure 1 is a graph set up to implement one call. Its edges are *dialogs*, while its nodes are *devices* and *feature boxes*. A *dialog* is an instance of a call-control signaling protocol. Each *feature box* is a concurrent software process that implements an instance of a feature.

A dynamic graph of dialogs and feature boxes is called a *usage* in DFC. All of the figures in this paper are examples of full or partial usages. Unlike Figure 1, most of the usages have subusages pertaining to multiple calls.

A usage such as Figure 1 composes its features because signals traveling from one device to another must pass through all of them. Each feature can modify these signals for its own purposes, so that the overall behavior of the usage reflects the functions of all of them. This is known as *pipes-and-filters composition.*

Figure 1 shows a usage implementing a single call from the person with telephone number p to the person with telephone number q. The subusage marked *src* $= p$ contains the feature boxes that are present because the call is from p. The subusage marked *dest* $= q$ contains the feature boxes that are present because the call is to q. All six features are call set-up features, which are introduced here because they will be used in examples throughout the paper.

This paper assumes that all features are implemented in the network, in application servers deployed for this purpose. Most likely each subusage of Figure 1 is implemented in a feature server assigned to its subscriber, so that the two subusages may be implemented in different servers, with a network connection between them.

Using an abstract call-control signaling protocol, a dialog begins when a box or device sends a *setup* signal to another box or device. The setup signal carries both source and destination telephone numbers. Once the *setup* has been acknowledged, the dialog exists, and its signaling channel can be used for commands, status signals, and control of media channels. A dialog can be torn down at any time, from either end.

The three most important status signals in a dialog protocol are *alerting, avail* and *unavail.* All of these signals travel from the callee or incoming end of the dialog to the caller or outgoing end. The alerting signal indicates that the person identified by the destination number is being notified of an incoming call, while *avail* indicates that he is available for communication. The dual of *avail* is *unavail*, which indicates that the person is not available.

A well-designed feature box has the properties of *transparency, autonomy,* and *context-independence. Transparency* means that when its feature is not active, it is unobservable by other boxes in the graph. It is acting as an identity element, merely relaying signals from one dialog to another. *Autonomy* means that when it needs to perform some function, it does so without help from other boxes. A feature box can act autonomously because it sits in a signaling path between user devices, where it can observe all the signals that travel between them. Because it is a protocol endpoint, it can absorb or generate any signals that it needs to. *Context-independence* means that it does not rely on the presence of other features, or contain any knowledge of them. A feature box does not know what is at the other end of the dialogs it is participating in.

In Figure 1, the source features of p are Speed Dialing (SD) and Outgoing Call Blocking (OCB). SD examines the setup signal to see if its destination field contains a short code rather than a telephone number. If so, it replaces the code with the telephone number that the subscriber has assigned to it. In all other respects SD is transparent.

OCB is configured by an authority other than the subscriber, for example a parent. OCB examines the setup signal to see if the subscriber is allowed to call that destination. If so, OCB is transparent. If not, OCB generates an error message for the subscriber and tears down its incoming dialog, which will cause the usage to be torn down all the way back to the originating device.

In Figure 1, the destination features of q are Incoming Call Blocking (ICB), Redirect to Voice Mail (RVM), Do Not Disturb (DND), and Parallel Ringing (PR). ICB blocks undesired calls as OCB does, but with two differences. First, ICB is usually configured by its own subscriber. Second, it blocks a call by sending *unavail* and then tearing down its incoming dialog; thus the caller cannot observe *why* his call did not succeed.

RVM is initially transparent. If it receives *unavail* through its outgoing dialog, then it tears down its outgoing dialog, creates another outgoing dialog whose destination is a voice-mail server, and becomes transparent. The server will accept the dialog (connecting it to the caller through RVM), send *avail* on behalf of the subscriber, and prompt the caller to leave voice mail for the subscriber.

If DND is currently disabled by the subscriber, the feature box behaves transparently. If it is currently enabled, its purpose is to allow only important calls, so that the subscriber is not otherwise disturbed. DND decides whether a call is important by taking the caller's word for it. Because DND cannot make any assumptions about the caller's device, its user interface to the caller must be implemented by audio announcements, prompts, and touch-tones.

On receiving a setup signal when it is enabled, DND employs a media resource (see Section 6) to announce to the caller that the subscriber wishes to be undisturbed. It then prompts to ask the caller (through a touch-tone response) if the call is important. If the call is not important, then DND sends *unavail* upstream and terminates, because the subscriber is not available for casual calls. If the call is important, then DND creates an outgoing dialog and is transparent from that point on.

PR creates concurrent outgoing dialogs to a list of phone numbers supplied by the subscriber, for example the numbers of the subscriber's home phone (qh) and work phone (qw). It also creates a dialog to the subscriber's mobile phone.

Note that a phone or other user device will send an avail signal upstream when the user answers the phone. If PR receives an avail signal from one of its downstream branches, it tears down the other branches and forwards the avail signal upstream. If it receives *unavail* from all branches or times out, it tears down all the branches, sends *unavail* upstream, and terminates.

2.2 Routing and feature interactions

The mechanism for assembling usages is a *feature router*. Each time a box initiates a dialog, the setup signal goes to a feature router that chooses a box or device to receive it, and forwards the *setup* to its recipient.

Every continuous routing chain from one device to another contains a *source region* and a *destination region*. The source region comes first; it contains feature boxes working on behalf of the source telephone number in its role as caller. The destination region contains feature boxes working on behalf of the destination telephone number in its role as callee. Each number *subscribes* to some (possibly empty) set of features in each region.

Figure 1 shows that p subscribes in the source region to SD and OCB, while q subscribes in the destination region to ICB, RVM, DND, and PR. In each region, the subscribed features of a telephone number have a *precedence* order that determines the order in which they are assembled into a usage.

A simple routing chain from device to device begins when the calling device creates a setup signal with the *new* method and sends it to a feature router. The source field of the *new* setup signal is the number of the device. To continue the chain, a feature box takes a setup signal it has received and applies the *continue* method to it. The *continue* method returns a setup signal, which the box then sends to a feature router. Each method returns a setup signal with the fields set so that a router can construct each region according to the precedence order.

If a feature is added to a subscriber's feature set, a new question arises: how should its signaling actions be merged and prioritized with respect to the signaling actions of the other features? This is the most common kind of feature interaction in a pipes-and-filters architecture. Once the desirable behavior has been determined, it is implemented by adjusting the precedence order of the feature set.

For instance, SD and OCB interact because SD changes the destination field of a setup signal and OCB examines the destination field of a setup signal. OCB should see the actual destination that will be called, so SD should precede OCB in a source region.

The interactions among the destination features of Figure 1 are analyzed in [14]. The precedence order shown yields behavior that satisfies the most common user expectations.

In the literature there are many other feature capabilities and ways of composing features, but they are not all equally useful. For instance, often features are described as sets of action rules on the call state with pre- and post-conditions. In these schemes feature composition is the union of the rule sets, and feature interactions are detected as conflicts between action rules from different features. As an example, [9] is unusual in including some mid-call actions, but typical in other ways.

Rule-based feature constructs have not had much impact on practice because the common ground of interaction between two features is the entire call state, which can become arbitrarily complex as features are added. The big advantage of pipes-and-filters composition is that the common ground of interaction between two features is only the dialogs between them, which grow little in complexity even in complex feature sets. Pipe-and-filters composition is constructive, and provides in precedence a simple mechanism for managing many feature interactions.

3. MID-CALL FEATURES WHERE JOINS OCCUR

3.1 Understanding the interactions

In this paper a *personal usage* is the portion of a usage containing all the features of a particular subscribing person. Starting at a time when the subscriber has no telephone activity, a personal usage is assembled when the subscriber first places or receives a call. Mid-call features allow the personal usage to change shape while the subscriber is talking. When the subscriber returns to a state of no telephone activity, the personal usage is torn down piece by piece and disappears.

A typical call set-up feature is subscribed to in one region only, because during call set-up the caller (source) and callee (destination) play very different roles. Mid-call features, on the other hand, operate when the caller/callee role distinction no longer matters much, and both ends of the call are symmetric. For this reason, mid-call features are subscribed to in both source and destination regions. If both the source and destination of a call subscribe to a mid-call feature, then the usage will have an instance of that feature in the source's personal usage, and another instance in the destination's personal usage.

One of the best-known mid-call features is call waiting (CW). CW is initially transparent, and its function is triggered only when it receives a new incoming call for the subscriber. It sends an alerting signal to the new call as if the subscriber's phone were ringing, and sends a signal to the subscriber that a call is waiting. On the subscriber's command, it will switch the subscriber back and forth between old and new calls. If CW receives another incoming call while it is already handling two calls, it will refuse the third call. It has long been known that CW interacts with unavailability features such as RVM, by narrowing the circumstances under which they are triggered [3].

Zooming out to look at a much less detailed picture, q's personal usage will look different when CW is active (Figure 2) than it does in Figure 1. Because CW is a mid-call feature, the subscriber has answered one device, and there is now only one outgoing dialog instead of three. More relevantly, CW allows a new incoming call to *join* an existing usage. In Figure 2 the usage has two dialogs coming in instead of one, each representing a different incoming call to q. These two incoming dialogs must share the single outgoing dialog that connects the usage to the mobile phone.

Any join feature raises two questions that must be answered before implementation can be completed:

- In the personal usage of q, how many instances are there of each of the other features?
- To which dialogs does each apply?

A join feature also requires that a single dialog (outgoing in

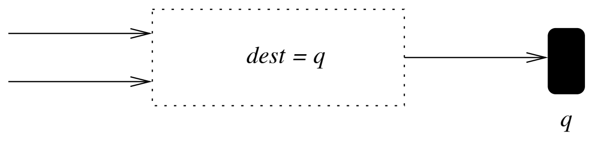

Figure 2: The personal usage of q when CW is active.

the figure) carry the signals of multiple dialogs (incoming in the figure). This may be inconsistent with the rules of the dialog protocol. These are the most important interactions caused by mid-call features where joins occur. The inconsistency will be dealt with in Section 3.2.

The obvious answer to the questions, for ICB and RVM, is: two, one applying to each of the incoming dialogs. PR, on the other hand, should only be applied when the first incoming call triggers the personal features to make contact with the subscriber. After PR has helped the subscriber choose and answer a device, the subscriber will remain connected on the same device while CW receives additional calls. So the right answers for PR are: one, applying to the outgoing dialog.

DND is interesting because it might be like ICB and RVM or like PR. Once the subscriber has been disturbed by a first incoming call, does DND allow subsequent incoming calls through? If so, it belongs in the same category as PR. Does DND continue to apply whether the subscriber has been disturbed by an urgent call or not? If so, it belongs in the same category as ICB and RVM. Note that the DND implementation remains exactly the same in either case, and only its place in the personal usage will be different.

If a subscriber has multiple join features, then different incoming calls can join a usage in different places, and therefore have different features applied to them. This means that the questions above apply separately to each join point. It also makes it necessary to distinguish the calls that should join in one place from the calls that should join in another.

As an additional simplification, this paper assumes that a personal feature set has only one join feature. The simplification is discussed further in Section 4.

The simplification does not mean that there can be only one join feature. Consider, for example, Emergency Break-In (EBI). EBI is a feature that allows an emergency call to break into an existing usage. The implementation of EBI is similar to the implementation of CW, except that it gives priority to the emergency call.

Emergency calls can be distinguished from normal calls because they are calls to *device* numbers, while normal calls are calls to *personal* numbers. Feature subscriptions are associated with telephone numbers. The personal features ICB, RVM, DND, and PR do not apply to emergency calls, so device numbers do not subscribe to them.

To provide for this behavior (and also to manage other feature interactions, as we shall see) it is necessary to have device numbers for mobile phones that are distinct from their personal numbers. So the mobile phones in subsequent examples will have device numbers *pm* and *qm*, while p and q remain as personal numbers. Device numbers need not be public.

3.2 Managing the interactions

This section describes how the desired behavior can be

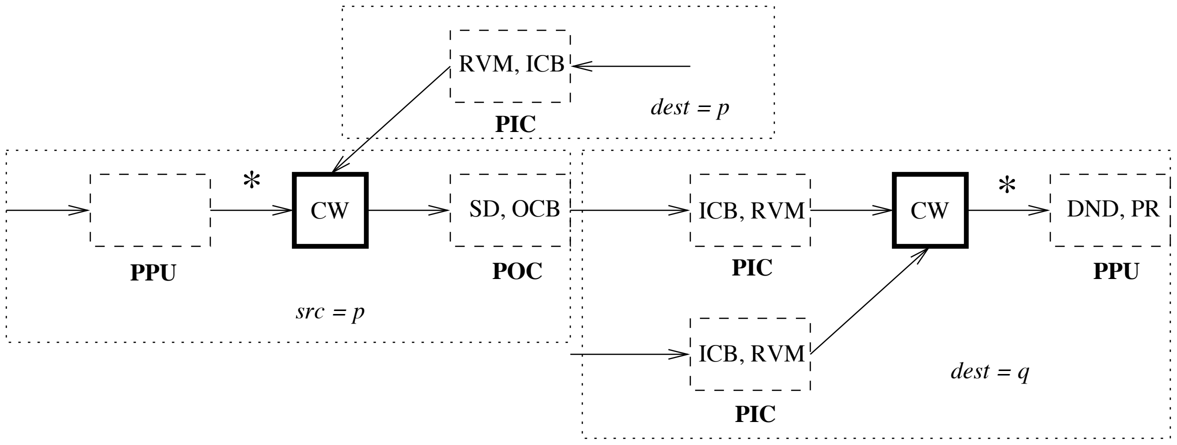

Figure 3: A mid-call join feature such as CW divides a personal usage into per-personal-usage (PPU), per-incoming-call (PIC), and per-outgoing-call (POC) subusages.

implemented using the advanced capabilities of DFC.

Returning first to CW and how joins occur, all the setup features in Figure 1 are *free*, meaning that when a feature router needs to route a dialog to a box of that type, it creates a new instance of the feature. Joins require features that are *bound* rather than free. If a feature is bound, then there is at most one box (instance) of that feature per subscriber. When a feature router needs to route a dialog to a box (instance) of that type for a particular subscriber, and there is already one in use for that subscriber, the router sends the dialog to the existing box. In Figure 3, CW is drawn with a heavier line to emphasize that it is bound.

In the personal usage of q found in Figure 3, CW divides the graph into three portions. There is a portion called the *per-personal-usage* (PPU) subusage, because each feature is instantiated only once in each personal usage. There are also two *per-incoming-call* (PIC) subusages, with features that apply separately to each incoming call, whether the entire personal usage is new or not.

The feature interactions are managed by setting the precedence order in q's destination region so that the PIC features precede CW and the PPU features follow it. As discussed above, ICB and RVM are PIC features. In this example DND is a PPU feature, as well as PR.

If the first call of the personal usage is outgoing, as is the case for subscriber p in the figure, then CW divides the early usage into a PPU subusage and a *per-outgoing-call* (POC) subusage. The POC subusage contains all the source features that apply separately to each outgoing call, whether the entire personal usage is new or not. The feature interactions are managed by setting the precedence order in p's source region so that the PPU features precede CW and the POC features follow it. In Figure 3 there are no features in p's PPU, and SD and OCB belong to p's POC.

A subsequent incoming call to p adds a PIC subusage to p's personal usage. Note that if there were a PPU subusage it would be assembled as a set of source (outgoing) features, yet its dialogs would be required to carry the signals of this incoming call.

Each PIC or POC subusage may contain a chain of dialogs, all of which represent the same call. A PPU subusage also has a chain of dialogs, but these dialogs may be *shared* among multiple calls. As introduced in Section 3.1, sharing may be inconsistent with the rules of the dialog protocol (it is inconsistent with SIP, as discussed in Section 8). Eliminating the inconsistency may require two modifications:

- It is usually necessary to represent call information (status, commands) differently in the shared and unshared portions of the personal usage. For example, an incoming dialog setup signal in a PIC or POC subusage turns into a special mid-call call-waiting signal in the PPU subusage.
- It is necessary to associate per-call signals in the PPU subusage with the calls to which they belong. Thus the call identifiers they carry may differ from the call identifier with which the subusage was set up, and must be hidden if this causes trouble.

All PPU features must handle shared dialogs. In practice this is not usually a burden. For example, the PPU features DND and PR are call set-up features, so they are triggered only by the setup signals of their initial incoming dialogs. After performing their functions they handle all mid-dialog signals transparently, which is correct whether the signals are shared or not.

In some cases the user interface to personal features employs a browser or other non-telecommunication application on the device. In this design, user-interface messages do not travel through the dialogs of a usage. To implement this design, it is necessary to associate each user-interface message to or from the device with the correct per-subscriber and per-call instance of the correct feature. Once it is implemented there are fewer dialog feature interactions, however, because the user interface is independent of the dialog protocol.

Figure 4 shows the relationship between personal features and EBI. Understanding this figure requires understanding the transitions between personal and device numbers.

A source feature can change the source number in its incoming dialog to a different source in its outgoing dialog, and a destination feature can change the destination number. (Features in each region can also change the opposite telephone number, but this is less important because it does not affect feature routing.) Figure 4 shows that PR changes destination q to qh, qm, and qw in its three outgoing dialogs.

If (as in the figure) qm subscribes to its own destination

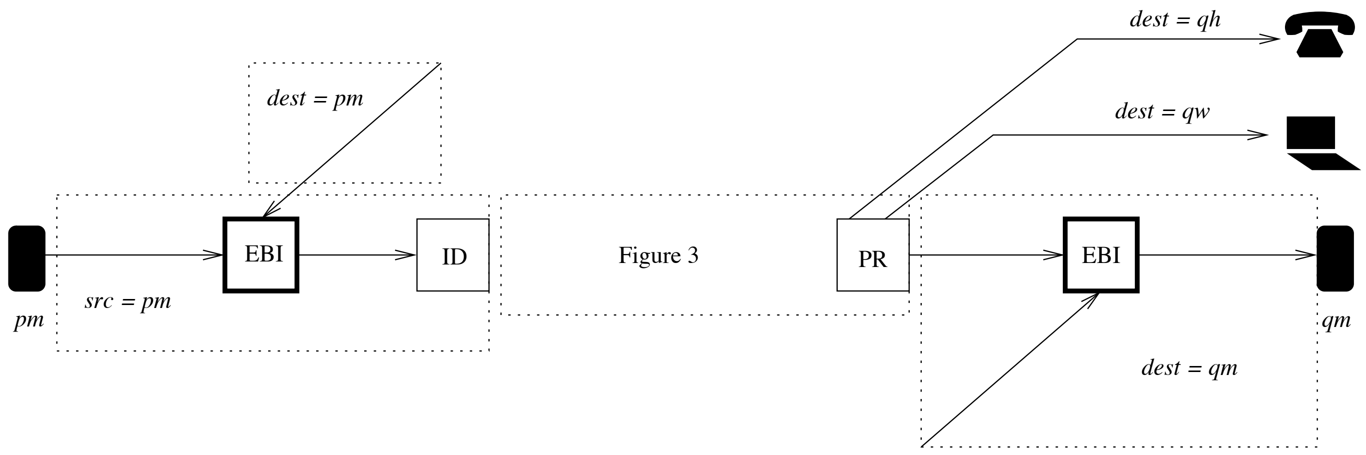

Figure 4: EBI is a mid-call join feature in a device zone rather than a personal zone.

feature EBI, then the chain from device *pm* to device *qm* will contain a destination *region* consisting of two destination *zones,* one for *q* and one for *qm.* Emergency (device) calls are given destination *qm* by their callers, so they will be routed directly to the zone of *qm*, without passing through the zone of *q.*

In Figure 4 the source region also has two zones, one for *p* and one for *pm.* In addition to EBI, the source zone for *pm* also contains the Identification (ID) feature, which changes source *pm* to source *p.*

Note that in more complex feature sets there can be multiple layers of phone numbers, multiple zones, and multiple join points in usages [12]. In this paper the discussion is simpler, because of the restriction to personal and device numbers only.

4. MID-CALL FEATURES THAT INITIATE NEW CALLS

Returning to CW for a few paragraphs, it is important to emphasize that a CW feature box, which is assembled into a personal usage during handling of the first call of a personal usage, may be routed to in either source or destination regions. For example, in Figure 3, the CW for *p* was routed to in the source region, and the CW for *q* was routed to in the destination region.

During initialization, a CW program must check the region of its first incoming dialog and orient itself properly. If the dialog is routed in the source region, then it comes from the subscriber, and its continuation goes toward a far party. If the dialog is routed in the destination region, then it comes from a far party, and its continuation goes toward the subscriber. In either case we refer to the dialog connecting CW to its subscriber as its *anchor* dialog. The anchor dialog will last as long as the personal usage does, while dialogs that connect CW to far parties can be set up and torn down more frequently. In Figure 3 the anchor dialogs of CW are marked with asterisks.

A personal feature set usually has a feature to initiate new outgoing calls while there is already a call in progress, so that conferences can be formed. The earliest form of this feature was Three-Way Calling (3WC).

3WC is exactly like CW in forming usages with a single dialog connecting the usage to a device, and multiple dialogs connecting the usage to far parties. Not surprisingly, it has the same feature interactions as CW, managed in the same way, as shown in Figure 5.

3WC has an additional feature interaction that CW does not have. Because CW initiates new calls, it raises the questions: What are the source numbers of these calls, and what personal features apply to them? In the personal usage of *p* this is fairly obvious—3WC always continues its anchor dialog, so that all its outgoing dialogs have the same source (*p*) and have the same features applied to them (the POC subusage).

For the 3WC feature of *q*, on the other hand, the answer is different, because this time 3WC is *reversing roles.* The source of the new outgoing dialog must be the destination of its anchor dialog (*q*). Even though the box was routed after PIC features in the destination region, its outgoing dialog must be routed through the POC features.

This behavior cannot be accomplished by continuing the anchor dialog, which is a destination-region dialog. To manage this feature interaction in DFC, we use DFC's third routing method, named *reverse.* The *reverse* method operates on the setup signal of an existing incoming or outgoing dialog, reversing its source/destination numbers and routing region. The effect of *reverse* applied to the anchor dialog of *q*'s 3WC is a setup signal that will be routed to SD, as the first feature in the POC subusage of *q.*

Note that, like *continue*, *reverse* allows a feature program to change the source or destination field in the new dialog. 3WC should not change the source field, because that is the subscriber's number. It must change the destination field, of course, to reach a new far party. Consequently the method invocation that creates the setup signal for 3WC's new dialog can be characterized abstractly as *reverse (dialog = anchor, dest = newFarParty).*

Because CW and 3WC are both *multi-call* features, they interact directly through the selection of calls they manipulate [6]. When implemented separately in DFC, their interactions show up as tree structures in a usage (Figure 6). The tree structure shows that the calls are divided into groups; for example, the subscriber cannot form a conference containing incoming call 1 and outgoing call 3. If this kind of

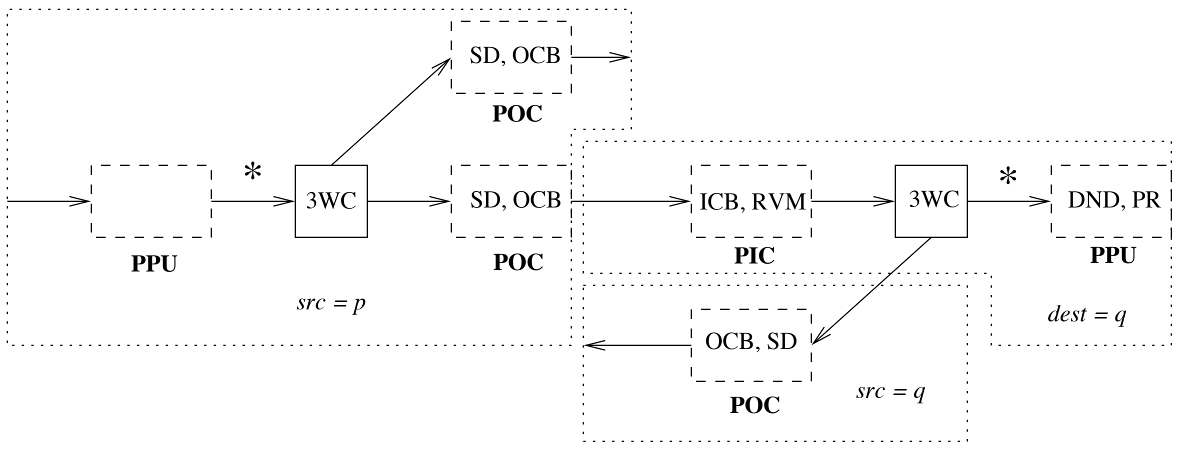

Figure 5: A call-initiating feature such as 3WC also divides a personal usage into per-personal-usage (PPU), per-incoming-call (PIC), and per-outgoing-call (POC) subusages.

grouping behavior is desired, then the tree structure is a good way to analyze and implement it.

To many people, Figure 6 seems wrong because we feel that all four calls are on an equal footing, and should be treated alike by the features. This is compatible with the assumption made in Section 3.1, that a personal feature set does not have a way of distinguishing different calls for different join treatment.

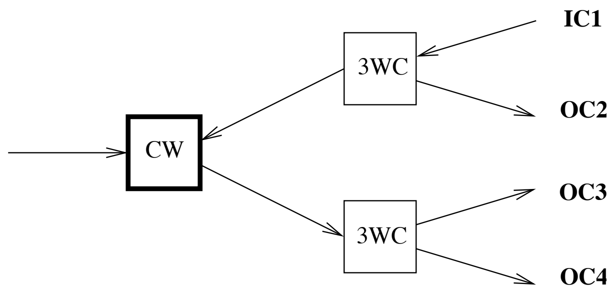

Figure 6: Interactions of multi-call features.

When a personal feature set includes joining and conferencing among calls regarded as equals, it is advisable to implement a single "multi-call control" feature that performs both functions. This leads to a better user interface and fewer constraints on what the subscriber can do. Not surprisingly, contemporary "smart" mobile phones typically combine CW and conferencing. They also handle more than four calls at a time, which would not be intelligible to users without the better interface and integration.

5. PER-PERSONAL-USAGE MID-CALL FEATURES

This section takes a closer look at the PPU subusage, dividing it into finer-grained subusages. We are working toward a final organizational structure for personal usages, which is shown in Figure 10. First we introduce two new mid-call features to be used as examples.

Recording (Rec) is a mid-call feature that records the voice channel (in both directions) whenever the subscriber desires. Rec must be placed in the PPU subusage so that it can capture everything that the subscriber hears and says.

Switch Phones (SP) is a mid-call feature that enables its subscriber to move from one device to another while maintaining a conversation. A well-designed SP feature creates a three-way conference during the transition, so that no part of the conversation is lost. SP must be placed in the PPU subusage so that all of the subscriber's calls are moved at once.

When it is activated, SP creates a new branch (chain of dialogs) to a new device that eventually takes over from the first or existing device branch. Because of this behavior, it raises many of the same questions as join and conferencing features, except on the device side of the personal usage instead of the network side:

- Which features should apply to the personal usage, independent of its device branches?
- Which features should apply to each device branch of the personal usage?
- What is the source telephone number of the new device branch?

Figure 7 shows the resolution and management of these feature interactions.

There is a subusage of a personal usage where the dialogs are shared by all device branches as well as all calls. This is called the *omniscient* (Omn) subusage, because features in it can observe all the end-to-end activity in the personal usage. In Figure 7 we put Recording in the omniscient subusage, so that it can record from all device branches. In the precedence order, omniscient features must precede SP in the destination region, and follow SP in the source region.

Features that apply to separate outgoing device branches must follow SP in the destination region. If there are any source features that apply to device branches, *i.e.,* to device branches coming into a personal usage, then they would precede SP in the source region.

The device branches created by SP are fundamentally sequential, even though two may co-exist for a brief handoff period. Not surprisingly, the first one demands different features. The features that apply only on the *first contact* with the subscriber in this personal usage are in the FC subusage,

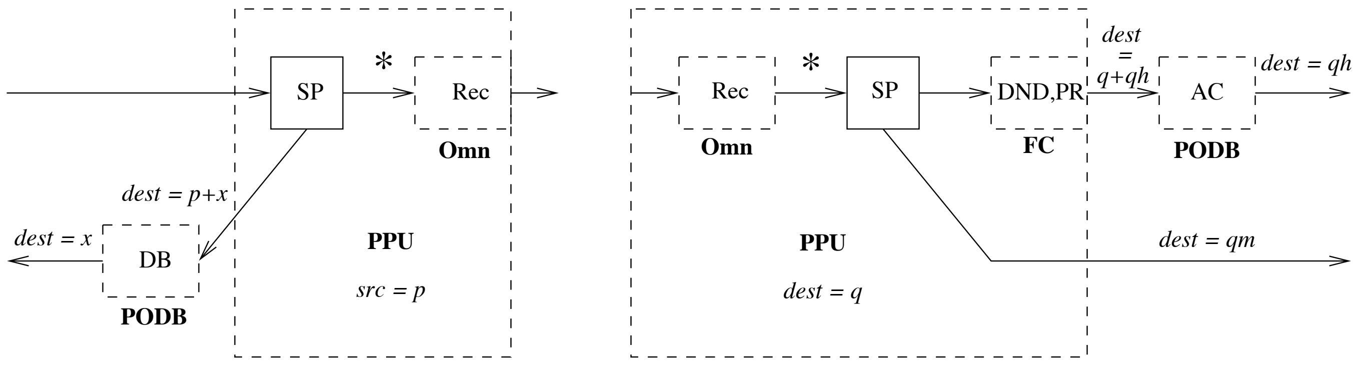

Figure 7: Per-personal-usage subusages of the personal usages of p and q. A multi-device feature such as SP divides them into omniscient (Omn) and first-contact (FC) subusages. Per-outgoing-device-branch (PODB) subusages are also introduced.

and those that apply to each device branch are in a *per outgoing device branch* (PODB) subusage, as shown in Figure 7. Note that PODB subusages are not contained within PPU subusages because—if a personal usage has several device branches—there may be more than one of a PODB feature per personal usage.

DND (as a PPU feature) and PR are first-contact features. In DFC, the application of these features to the first device branch only is managed as follows. Because SP routed to in the destination region always continues its initial dialog transparently, the first outgoing dialog from q's SP will have q as its destination, and be routed through DND and PR. In Figure 7, the incoming call was answered on q's home phone with device number *qh*. When subscriber q wants to switch phones, on the other hand, SP sets the destination of its new outgoing dialog to device number *qm* rather than personal number q. This branch is not routed through the FC features of q.

Just like a 3WC feature box routed to in the destination region, an SP feature box routed to in the source region *reverses roles* when it creates a dialog to begin a new outgoing device branch. As with 3WC, the SP feature must *reverse* its anchor dialog to create a dialog setup signal that will be routed correctly. The anchor dialogs of the SP boxes are marked with asterisks in Figure 7. 3WC and SP are symmetric features in the sense that 3WC creates new calls (going toward the center of the network) and SP creates new device branches (going toward devices on the periphery of the network). This symmetry is the reason why role reversal is performed by 3WC routed to in the destination region and SP routed to in the source region, which is also symmetric.

To create a dialog setup signal that will be routed correctly, SP routed to in the source region does a method invocation that can be characterized abstractly as *reverse (dialog = anchor, dest = newDevice)*. The source field of the setup signal will have the same value as the destination of the anchor dialog. The feature can also be implemented to use *reverse (dialog = anchor, dest = newDevice, src = subscriber)*, if it is considered desirable to identify the new branch as coming from the subscriber. Because the new dialog will be routed in the destination region, the source field will be used for caller identification only.

It may be necessary to choose specific PODB features for specific devices. Figure 7 shows two examples of this.

Answer Confirm (AC) is a feature that distinguishes between a call's being answered by a person and being answered by a machine. It does this by prompting the callee to enter a touch-tone. AC then signals *avail* or *unavail* upstream through its incoming dialog, depending on whether it detected the tone or not.

In the example, AC is necessary on outgoing device branches to the home phone *qh*, because q has PR and the home phone has built-in voice mail. Without AC, voice mail might answer the home phone and cause PR to abort branches to other devices. This might prevent the subscriber from answering the call on another device, or it might cause voice mail to be recorded on the home phone rather than the subscriber's personal voice mailbox.

On the left side of Figure 7, subscriber p is using SP to switch to a number x previously unknown to the system. Lest this be used by the subscriber to surreptitiously transfer a far party to a forbidden number, the outgoing device branch is routed through a Device Blocking (DB) feature that checks the number.

Because of the necessity to distinguish between FC and non-FC features, these PODB cannot be subscribed to by the personal numbers p and q. They are not (in general) subscribed to by the device numbers *qh* and x, because the features associated with these devices may have nothing to do with p or q. In DFC they are subscribed to by special internal numbers, denoted here as $q{+}qh$ and $p{+}x$ to show that they are related to both device and personal numbers. These numbers are internal mechanisms whose only purpose is to cause the DFC routing algorithm to assemble the desired features into personal usages.

SP is not the only possible multi-device feature. Another possible feature in this category is an Add Video (AV) feature that creates an outgoing device branch of the usage to a video device such as a television or laptop computer. This feature could enable a subscriber to upgrade a call on a voice-only device to a multi-media call.

Just as separate join and conferencing features create tree structures in a usage (Figure 6), separate multi-device features create tree structures (Figure 8). For multi-device features to interact correctly, it is necessary to understand the desired shape of the tree and set the precedence order so that routing creates the desired shape.

Section 4 recommended merging multi-call features into

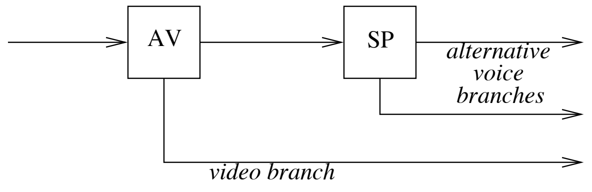

Figure 8: Interactions of multi-device features.

a single feature, because that facilitates treating all calls as equals. The same recommendation does not apply to multi-device features, because multiple device branches are likely to be functionally distinct. The structure of Figure 8, for instance, supports treating voice and video devices differently.

6. MEDIA FEATURE INTERACTIONS

Many features are implemented using media resources. For example, 3WC and SP require mixers, Recording requires a recording server, and DND and AC require interactive voice-response servers. These features have to ensure that media paths travel to the correct resources at the correct time. CW might be implemented with a resource to play special tones and announcements. Even if it is not, CW must still manipulate the media paths between devices.

All mid-call features that manipulate media paths have the potential to interact, because at any moment, the correct configuration of media paths might be dependent on the states of more than one feature. Figure 9 illustrates this. In the figure the signaling path between the mobile phone and the soft phone on the laptop contains two feature boxes, Rec and 3WC. Each feature is implemented with its own reasource, which is also shown in the diagram.

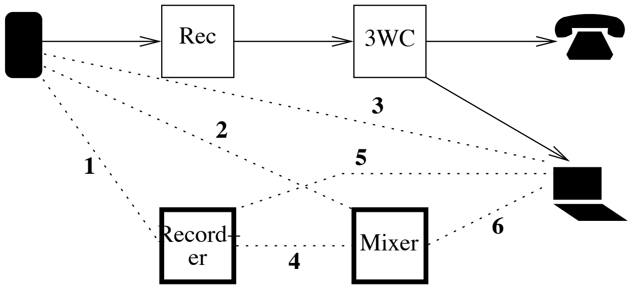

Figure 9: Media feature interactions affect the paths of media packets.

If neither feature is currently active, voice between the mobile phone and laptop should travel by the most direct path, labeled 3. If Recording alone is active, voice should travel on path 1, 5. If 3WC alone is active, voice should travel on path 2, 6. If both features are active, voice should travel on path 1, 4, 6. It is not possible to configure the voice paths without taking both features into account.

This example shows us that composition of media-manipulating features raises the new question: If multiple features in a usage are manipulating media paths concurrently, what are the correct end-to-end media paths? In [16] the proposed answer is based on the principle that, among composed features, proximity to an endpoint (in the usage graph) confers priority in determining its media paths. This leads to a general specification of the correct media paths in each case, and a general-purpose implementation of the specification that has been verified correct [16].

In further work, this general solution has been adapted for implementation in SIP [5]. A convenient programming interface to the SIP solution is described in [15].

7. SUMMARY

Figure 10 summarizes Sections 3 through 5 by providing a single picture of all the subusages of a personal usage.

Multi-call and multi-device mid-call features interact with all other personal features by forcing decisions about which calls and which device branches they apply to. These decisions affect how often a feature is instantiated, exactly when a feature is instantiated and activated, and exactly which subset of the set of all signals it sees. Although DFC makes it easy to analyze these interactions through the shapes of usages, DFC does not create them. They are intrinsic to the functions of mid-call features, and it is necessary to understand and manage them no matter how a feature set is implemented.

The other interactions caused by mid-call features are:

- Multi-call and multi-device features may cause some dialogs to be shared among multiple calls and/or device branches. This introduces new signaling information that must be encoded so that it is accepted by the dialog protocol and all feature boxes and devices at the endpoints of the dialogs.
- Multi-call and multi-device features may reverse roles, raising new questions about the source and destination numbers that should be used after role reversal.
- Multi-call features interact with each other by grouping calls in specific ways.
- Multi-device features interact with each other by grouping device branches in specific ways.

In addition, as previously reported, all mid-call features that manipulate media streams have the potential to interact with each other in determining correct media paths.

DFC includes some advanced capabilities that make it easier to manage these feature interactions. These advanced capabilities are: a flexible protocol that accommodates shared dialogs, bound features, the *reverse* routing method, and internal telephone numbers. Section 8 discusses the implementation of these capabilities in SIP frameworks.

Figure 10 shows that personal features are naturally divided into eight categories. This paper has given specific examples of features in all categories except *per-incoming-device*. In practice there can be any number of features in these categories. There are also other features that, like DND, can go into more than one category, and produce different overall behavior depending on their category. Another example is Talk Limits (TL), which limits the number of telephone minutes a subscriber is allowed. TL is a mid-call feature that can be omniscient or per-call; depending on where it is placed in a personal usage, it measures and limits different things.

In the context of converged Web and telecommunication services, there can be a dialog-state notification service, and other services that subscribe to the dialog-state notifications for various purposes [10]. The analysis of mid-call features here shows that the relevant state is actually the state of a personal usage rather than of any particular dialog, and

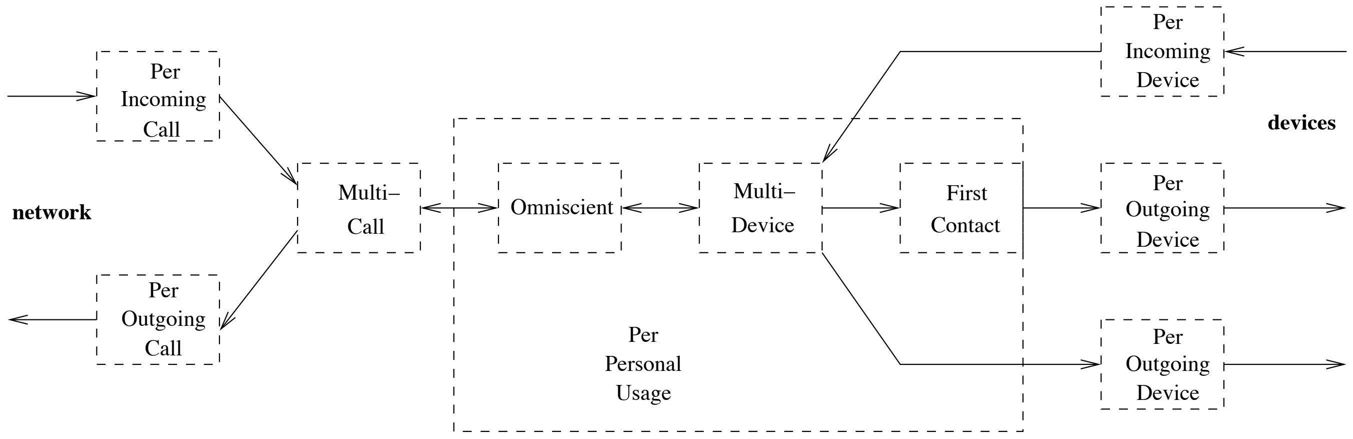

Figure 10: All the subusages of a personal usage. Dialogs with double-headed arrows can be set up in either direction.

that the state of a personal usage is much more complex than a dialog state. Thus mid-call features interact with both the notification service and the converged services that subscribe to it.

Click-to-Dial (C2D) is always an outlier in the feature landscape. In its most common form, C2D creates two outgoing dialogs (whether sequentially or simultaneously), both routed to destination features. It thus creates a usage with two personal usages, each first assembled with destination routing. The personal usages are themselves normal in all respects, however.

DFC can be used to develop feature sets with fine-grained modularity. In this style, each named function, no matter how simple, is implemented as a separate feature box. Although this style is good for prototyping because it is very flexible, it may incur too much overhead for production use [4].

To improve performance when using DFC principles, it is often desirable to group more functions into larger feature modules. Figure 10 provides important guidance in doing this, because a successful coarse-grained modularization must be compatible with its structure. The most obvious coarse-grained modularization has one feature box per atomic subusage in Figure 10.

Even coarser-grained modularizations are possible, provided that they coalesce adjacent subusages and preserve the correct instantiations. For example, it is straightforward to combine omniscient functions with multi-call or multi-device functions. It is possible to combine per-call functions with multi-call functions, but the resultant module must have an internal mechanism to instantiate per-call features for each new call. It is inadvisable to combine omniscient functions with first-contact functions, on the other hand, because it is impossible to compose such a module successfully with multi-device features.

8. IMPLEMENTATION

Assuming that feature interactions in a feature set are being managed according to DFC principles, it will be necessary to implement the advanced capabilities of DFC—those beyond simple pipes-and-filters as described in Section 2. This section covers the relevant implementation issues for SIP, SIP Servlets, and IMS.

8.1 SIP

In a SIP implementation, all the dialogs mentioned above would be *invite dialogs*. A dialog-setup signal would be a SIP *invite*, an alerting signal would be a *180*, an avail signal would be a *2xx* response to the initial *invite*, and an unavail signal would be a *4xx-6xx* response to the initial *invite*.

Shared dialogs are one of the biggest effects of multi-call and multi-device features. The SIP standard is not written to accommodate shared dialogs, so that the obvious implementations of them are inconsistent with the protocol. For example, a dialog endpoint cannot send *180* after sending a *2xx* response to the initial invite, even though shared dialogs often need to signal "ringing" mid-call. Some techniques for implementing shared dialogs in SIP are presented in [13].

8.2 SIP Servlets

The most recent SIP Servlet standard [8] is very compatible with the DFC architecture. A servlet corresponds to a feature. An application session corresponds to a feature instance or box. The session key-based targeting mechanism supports the distinction between free and bound boxes. The regions/roles are named *originating* and *terminating* instead of *source* and *destination*.

The standard allows compliant servlet containers to run application (feature) routers chosen by their deployers. For those who wish to deploy the DFC routing algorithm, an open-source DFC application router is available at `echarts.org`.

Reverse routing has long been a source of confusion, and its history reflects this. The DFC manual says that only an *outgoing* dialog can be reversed [17]. The SIP Servlet standard says that only an *incoming* dialog can be reversed.

In principle, the incoming/outgoing distinction should make no difference to an implementation. The only necessary restriction on the use of *reverse* is that the subscriber must subscribe to the reversing feature in both regions, so that the feature has a place in the precedence order of both regions. In practice, an incoming/outgoing restriction is simply a consequence of other implementation decisions.

Unfortunately, the incoming/outgoing distinction does make a difference in the arenas of requirements and behavior. As has been explained in Sections 4 and 5, 3WC and SP should reverse their anchor dialogs because they are the predictable and persistent dialogs. In both the cases where *reverse* is needed, their anchor dialogs are *outgoing* (see Figures 5 and 7).

Furthermore, consider the behavior of p's SP box (Figure 7) after it has switched to device x. After the switch is complete, its original incoming dialog will be torn down. If the box is called upon to switch phones again, at that time it will have *no* incoming dialog, whether anchor or not.

To overcome this deficiency in the SIP Servlet standard, an SP box routed to in the source region must save the setup signal of its original incoming dialog, even after that dialog has been torn down. When it needs to create a new device branch, it can apply the *reverse* method to the saved incoming setup signal.

8.3 IMS

The IP Multimedia Subsystem (IMS) standard builds on SIP [1] and is compatible with application servers programmed using the SIP Servlet standard. Like the servlet standard, IMS recognizes originating and terminating regions. An IMS S-CSCF is capable of routing a SIP call through one or more application servers in the originating region, to implement originating features, and one or more application servers in the terminating region, to implement terminating features. Like a DFC feature router, an IMS S-CSCF can create ordered chains of SIP dialogs and application servers.

If a subscriber's personal feature set contains a bound feature subscribed to in both regions, then all dialogs routed to that feature for that subscriber, whether in the originating or terminating region, must go to or within the same application server. This is an additional requirement on application routing, not mentioned in the standard.

IMS puts subscriber data in separate databases with many administrative constraints. This may make it difficult to use internal telephone numbers to manage feature interactions, as done in Section 5. Dialogs routed to internal telephone numbers are best confined to within SIP Servlet containers, where these numbers can be configured locally.

9. CONCLUSION

This paper has explored many previously unreported feature interactions caused by mid-call features, particularly multi-call and multi-device features. The discussion has been focused and simplified by assuming a personal feature set associated with a mobile telephone number.

During an interval of time when a subscriber has telecommunication activity, a personal usage encompasses the network state of that activity. A personal usage may include the states of multiple incoming and outgoing calls, the state of the subscriber's connection with the network through one or more devices, and the states of relevant features. Because all of these elements of a personal usage can interact, an implementation architecture that represents these elements and their relationships is more robust than one representing calls alone.

The DFC architecture is well-suited to representing personal usages, It is used in this paper to analyze feature interactions and manage them by dividing personal features into eight categories. Because DFC does not create these interactions, all implementations of telecommunication services with mid-call features will have to manage them one way or another.

Acknowledgments

This work has benefited greatly from the contributions of my AT&T colleagues Greg Bond, Eric Cheung, and Tom Smith.

10. REFERENCES

[1] 3GPP. Service requirements for the IP multimedia core network subsystem. 3GPP Technical Specification 23.228 Stage 2.

[2] G. Bruns. Foundations for features. In *Feature Interactions in Telecommunications and Software Systems VIII*, pages 3–11, Amsterdam, 2005. IOS Press.

[3] E. J. Cameron, N. D. Griffeth, Y.-J. Lin, M. E. Nilson, W. K. Schnure, and H. Velthuijsen. A feature-interaction benchmark for IN and beyond. *IEEE Communications*, 31(3):64–69, March 1993.

[4] E. Cheung and T. M. Smith. Experience with modularity in an advanced teleconferencing service deployment. In *Proceedings of the Thirty-First International Conference on Software Engineering.* IEEE, 2009.

[5] E. Cheung and P. Zave. Generalized third-party call control in SIP networks. In *Proceedings of the Second International Conference on Principles, Systems and Applications of IP Telecommunications*, pages 45–68. Springer-Verlag LNCS 5310, 2008.

[6] Y. Inoue, K. Takami, and T. Ohta. Method for supporting detection and elimination of feature interaction in a telecommunication system. In *Proceedings of the International Workshop on Feature Interactions in Telecommunications Software Systems*, pages 61–81. IEEE Communications Society, 1992.

[7] M. Jackson and P. Zave. Distributed Feature Composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering*, 24(10):831–847, October 1998.

[8] JSR 289: SIP Servlet API Version 1.1. Java Community Process Final Release, `http://www.jcp.org/en/jsr/detail?id=289`, 2008.

[9] A. F. Layouni, L. Logrippo, and K. J. Turner. Conflict detection in call control using first-order logic model checking. In *Feature Interactions in Telecommunications and Software Systems IX*, pages 66–82, Amsterdam, 2008. IOS Press.

[10] X. Wu, J. Buford, K. Dhara, M. Kolberg, and V. Krishnaswamy. Feature interactions between Internet services and telecommunication services. In *Proceedings of the Third International Conference on Principles, Systems and Applications of IP Telecommunications.* ACM SIGCOMM, 2009.

[11] X. Wu and H. Schulzrinne. Handling feature interactions in the Language for End System Services. In *Feature Interactions in Telecommunications and Software Systems VIII*, pages 270–287, Amsterdam, 2005. IOS Press.

[12] P. Zave. Address translation in telecommunication features. *ACM Transactions on Software Engineering and Methodology*, 13(1):1–36, January 2004.

[13] P. Zave. Audio feature interactions in voice-over-IP. In *Proceedings of the First International Conference on Principles, Systems and Applications of IP Telecommunications*, pages 67–78. ACM SIGCOMM, 2007.

[14] P. Zave. Modularity in Distributed Feature Composition. In B. Nuseibeh and P. Zave, editors, *Software Requirements and Design: The Work of Michael Jackson*, pages 267–290. Good Friends Publishing, 2010.

[15] P. Zave, G. W. Bond, E. Cheung, and T. M. Smith. Abstractions for programming SIP back-to-back user agents. In *Proceedings of the Third International Conference on Principles, Systems and Applications of IP Telecommunications*. ACM SIGCOMM, 2009.

[16] P. Zave and E. Cheung. Compositional control of IP media. *IEEE Transactions on Software Engineering*, 35(1), January/February 2009.

[17] P. Zave and M. Jackson. The DFC manual. Technical report, AT&T Research, 2003. `http://www2.research.att.com/~pamela/man.pdf`.