Using Software Architecture to Aid in the Feature Interaction Problem

Ann Zimmer
School of Computer Science
University of Waterloo

Features and Services

- A service is a basic system, which provides standalone functionality. (i.e. basic phone service)
- A feature provides extra functionality to an existing service or feature, it can not be used on its own. (i.e. voice mail)

Feature Interaction

A feature interaction occurs when two or more features run simultaneously in a system and cause uncertain output. (i.e. the execution of one feature interferes with the output of another.)

- We will consider feature interactions to occur in 2 situations:
  - A feature adds a predicate that another feature has removed (or vise versa)
  - A feature tries to modify an unique identifier. For example a user can only hear one output message at a time.

Feature Interaction

- Call Waiting Verses Voice Mail
  - Call Waiting(CW): when user A subscribes to CW and is on the phone, a CW tone is issued to alert A of another incoming call.
  - Voice Mail(VM): when A subscribes to VM and is on the phone, any incoming calls are forwarded to a voice messaging system.
- What happens when A subscribes to both CW and VM?

CW verse VM

- Call Waiting

```
if state(TCM allocate c2 t o)
 and state(POTS voice c t o)
 and
  not (state(CW ringback * t
    o))
 and not (= c2 c)
 and not (cw_hold (* t))
 and CW(t)

then
   %state(TCM present c2 t o)
 and +connection(c2 t)
 and + state(CW active c c2)
 and +CW_hold(c2 t)
```

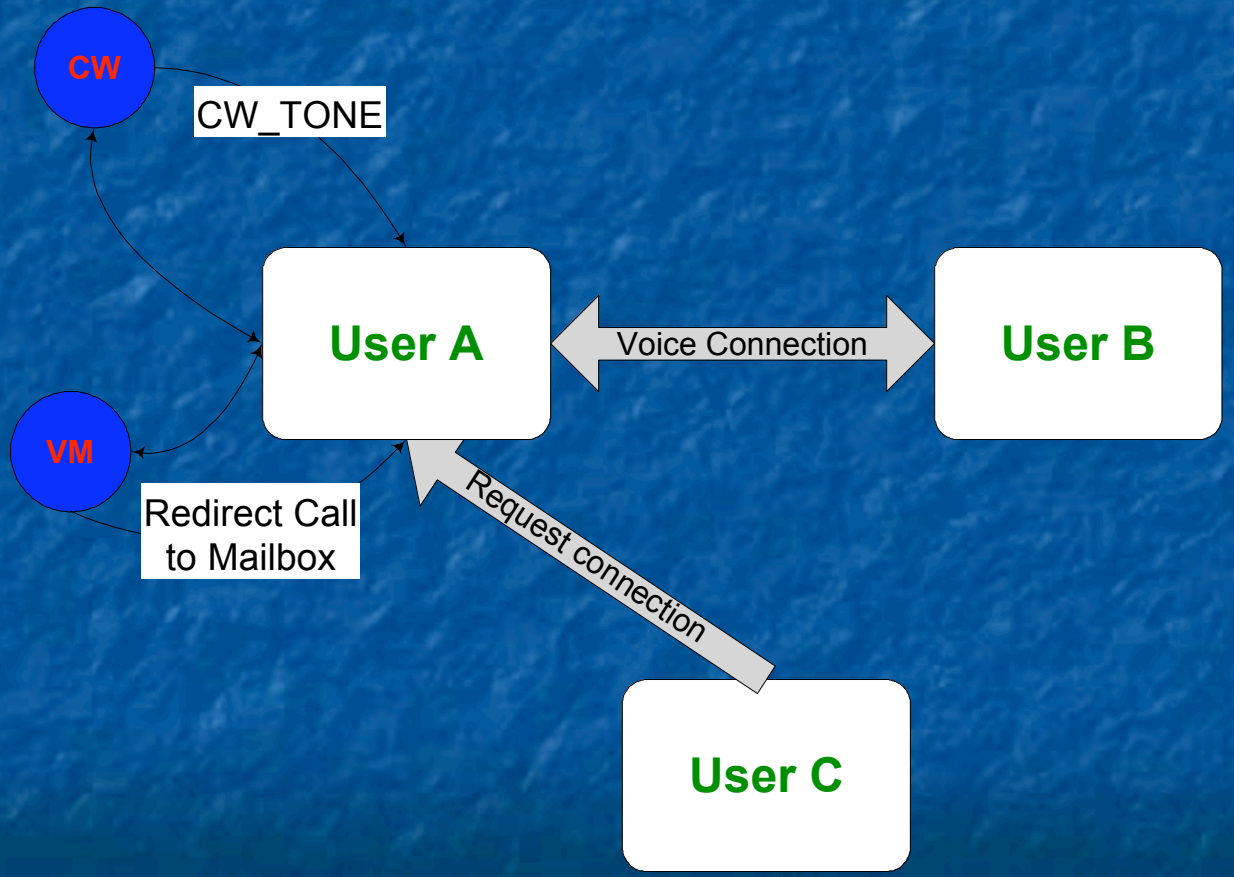
- Voice Mail

```
if state(TCM allocate c t o)
 and connection(c2 t)
 and (not (= c2 c))
 and VM(t)



then
   -state(TCM allocate c t o)
 and -connection(c t)
 and
   +msg(redirect o c t vm)
```

Feature Interaction:
Call Waiting verses Voice Mail



Outline

- Telephony and Architecture
- COURT:Co-Ordinating User-preferences at Run-Time
  - Coordination Model/Blackboard
  - Feature interaction resolution
  - Overview of benefits
- DFC: Distributed Feature Composition
  - Pipe and Filter Architecture
  - Feature interaction resolution
  - Overview of benefits

Current Telephony Architecture

- "Million if statements" – each feature is aware of all other features and resolution is hard-coded using if statements.
- We want to move away from this resolution strategy to allow for quicker development of new features.

Software Architecture

- Many of the existing methods to solve the feature interaction problem focus on changes to the system's architecture, which could help prevent or resolve certain types of feature interaction problems.
- The different choices for architectural design can influence the difficultly of solving the feature interaction problem.
  - Pipe and Filter (DFC)
  - Coordination Models (COURT)

Previous Research: COURT

- COURT: Co-Ordinating User-preferences at Run-Time
- Features are developed modularly (separation of concerns), so that each feature runs independently of the core system, without any knowledge of other features.

Coordination Model

- "A coordination model is the glue that binds separate activities into an ensemble" [Gelernter and Carriero]
  - Blackboard
  - Client-Server
- Coordination models are used to monitor processes in parallel and distributive systems

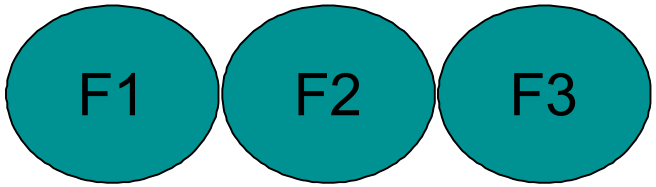
Blackboard

- Two components:
  - A central data store
  - Collection of independent components.
- The components work on the information found in the data store.
- Also known as a repository.

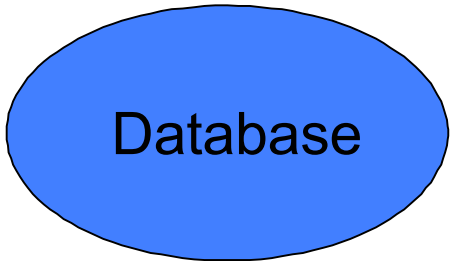
COURT

- COURT = Co-Ordinating User-preferences at Run-Time.
- COURT's purpose is to support modular design of features and to resolve interactions as they occur.
- Features are developed independently and extend the functionality of the core system.

COURT Components



Independent feature modules



Database of the current system state

FIM
Interaction & Resolution

FIM is designed to resolve interactions according to a given technique

COURT Features

- A feature is represented as a set of rules that are applied at different stages.
- Each rule has guard and action conditions.

on *event*
(message)
if *guard condition*
(predicate on facts)
then *actions*
(add or remove facts)
assert *constraints*
(predicates on facts)
retract *constraints*

COURT Database

- Feature (Voice Mail)

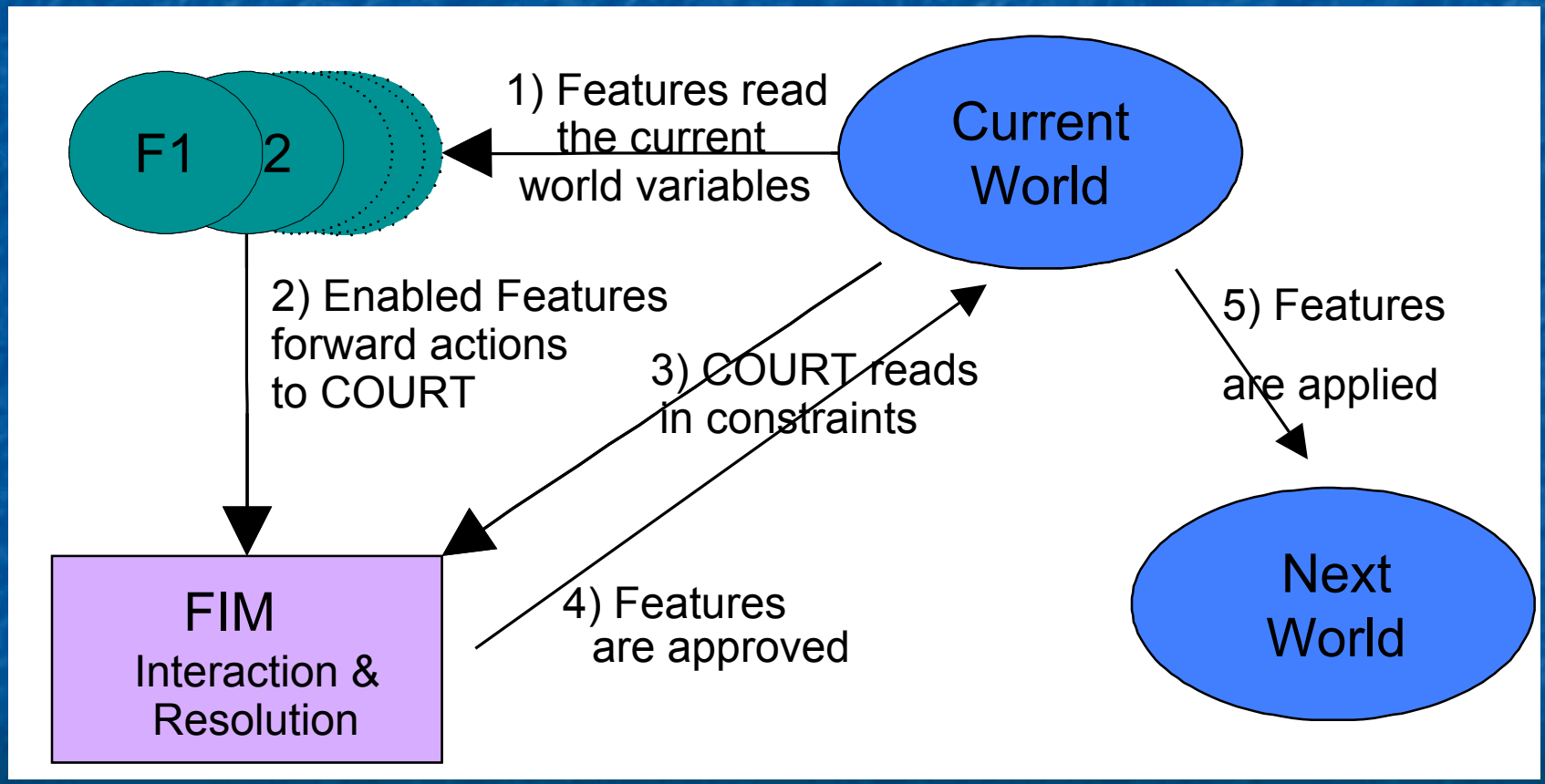
if *state(TCM allocate c t o)*
and *connection(c2 t)*
and *(not (= c2 c))*
and *VM(t)*

then
-state(TCM allocate c t o)
and *-connection(c t)*
and *+msg(redirect o c t vm)*

- Sample predicates

User(bob)
User(jamie)
User(sally)
VM(bob)
CW(bob)
3WC(sally)
Constraint(OCS sally bob)
connection(1 sally)
connection(1 jamie)

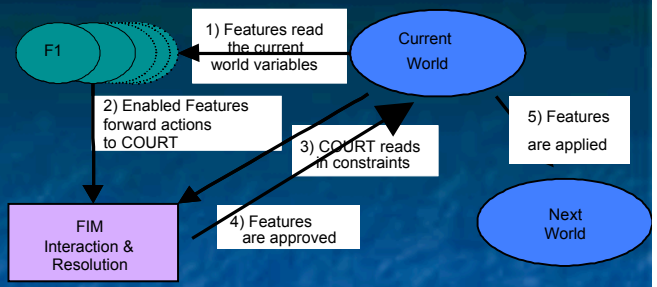
COURT Architecture



COURT Process

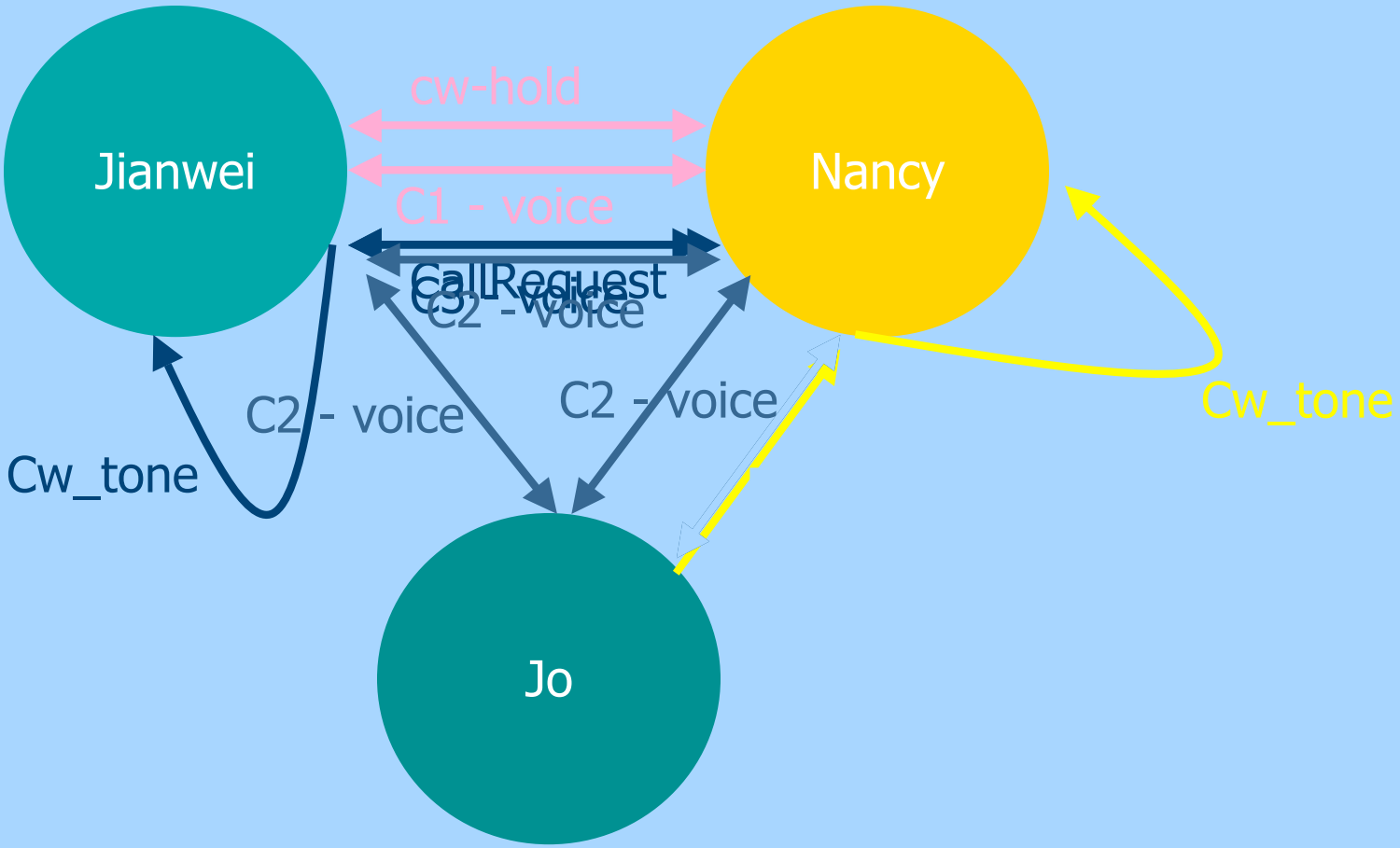
- Identifies the set of rules in the system.
- Read the current world.
- Identify which rules have their guard conditions satisfied.
- Analyze this set of rules for feature interactions.
- Resolve interactions.
- Determine the next world.

Feature Interaction Resolution



- Features are applied in order of priority
- After each feature is applied the resulting world is tested for interactions
- If an interaction is found the feature is rolled back and will not be executed.

Example: CW vs. 3WC



COURT advantages

- Separation of Concerns
- Less Coupling
- Features can be added by outside parties.
- Reduce rewriting of existing features.
- System will run consistently
- Ability to allow user defined features

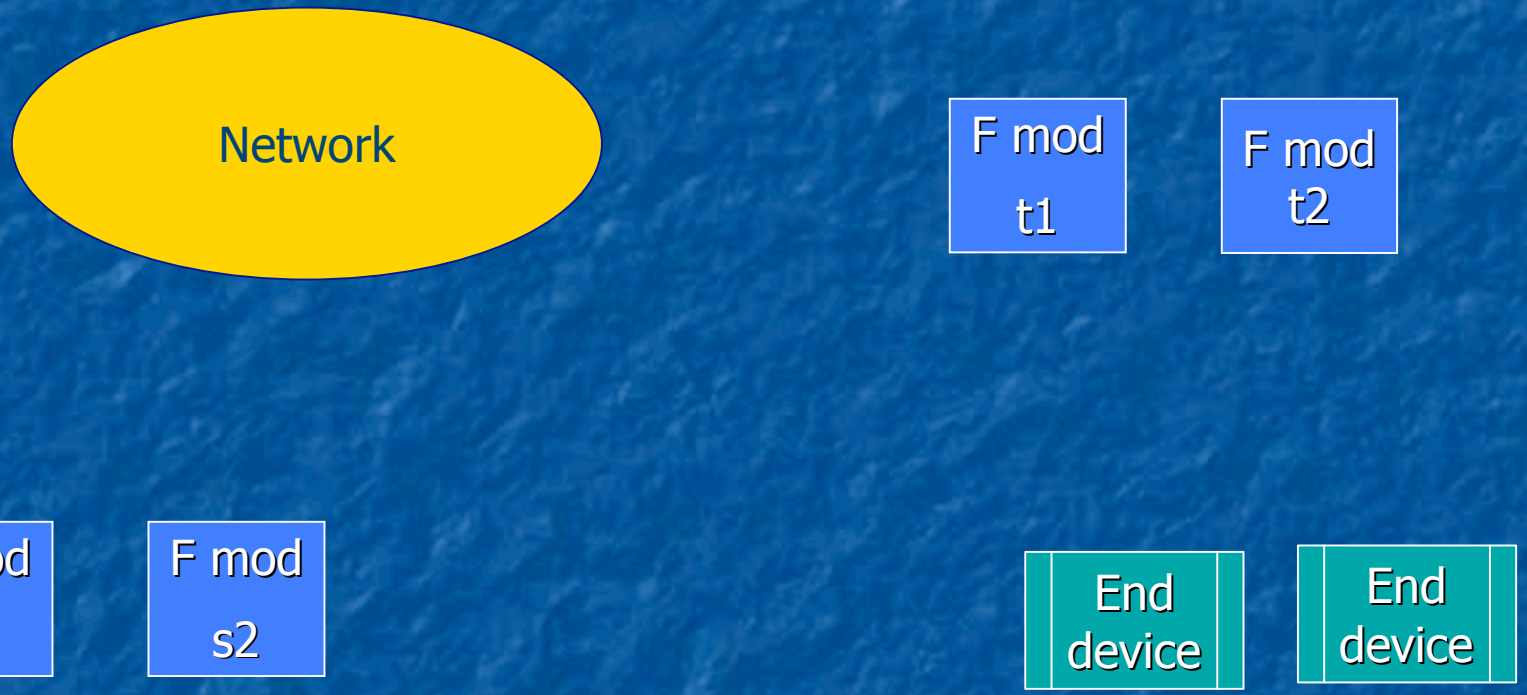
Current Research: Distributed Feature Composition (DFC)

- We have recently started working on a research project in association with AT&T. Work on the feature interaction problem is being explored with respect to DFC.
- DFC is the virtual architecture used by AT&T, where each feature is implemented as an independent feature box.

Pipe and Filter

- In a pipe and filter system there are two main types of components that are responsible for reading and transforming data.
  - Pipe: Is responsible for passing information from one filter to another.
  - Filter: Is responsible for reading in data and transforming the information before passing along the modified information.

DFC Components



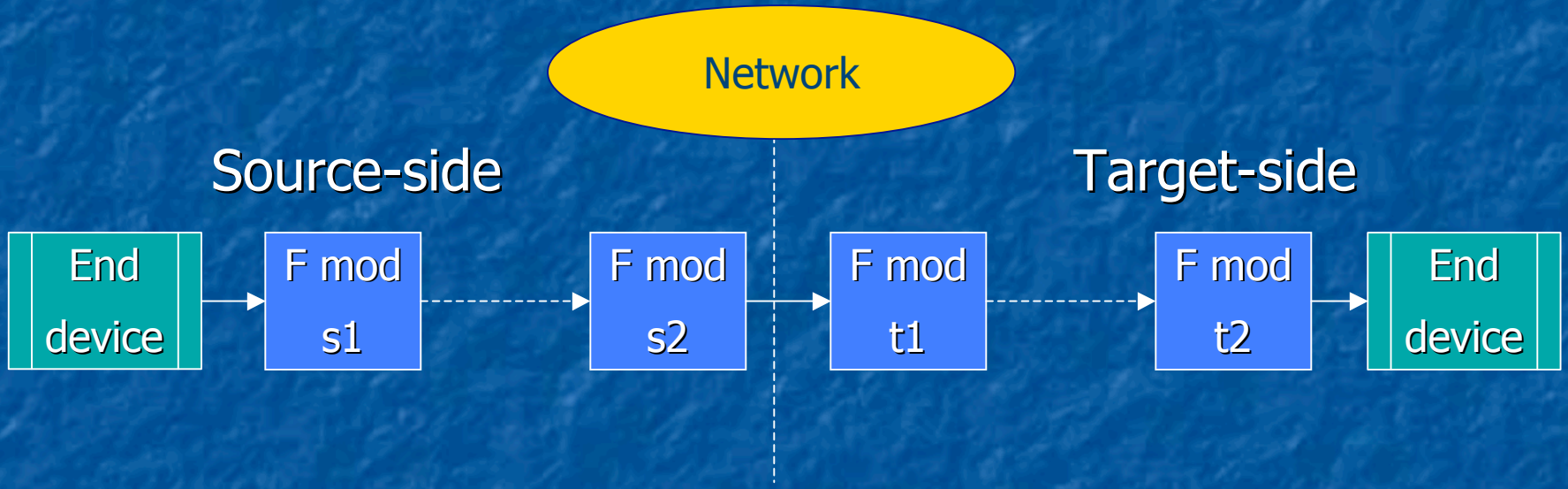
DFC Features

- Each feature module is designed independently.
- A feature module can be a source feature, a target feature, or both.
- The feature responds to the incoming signals and can modify, absorbs, or pass the incoming signal along. New signals may also be generated.

DFC Network

- The networks responsibility in DFC is to determine which feature modules appear in the call set-up.
- Some global information may also be stored in the network database.

DFC Architecture

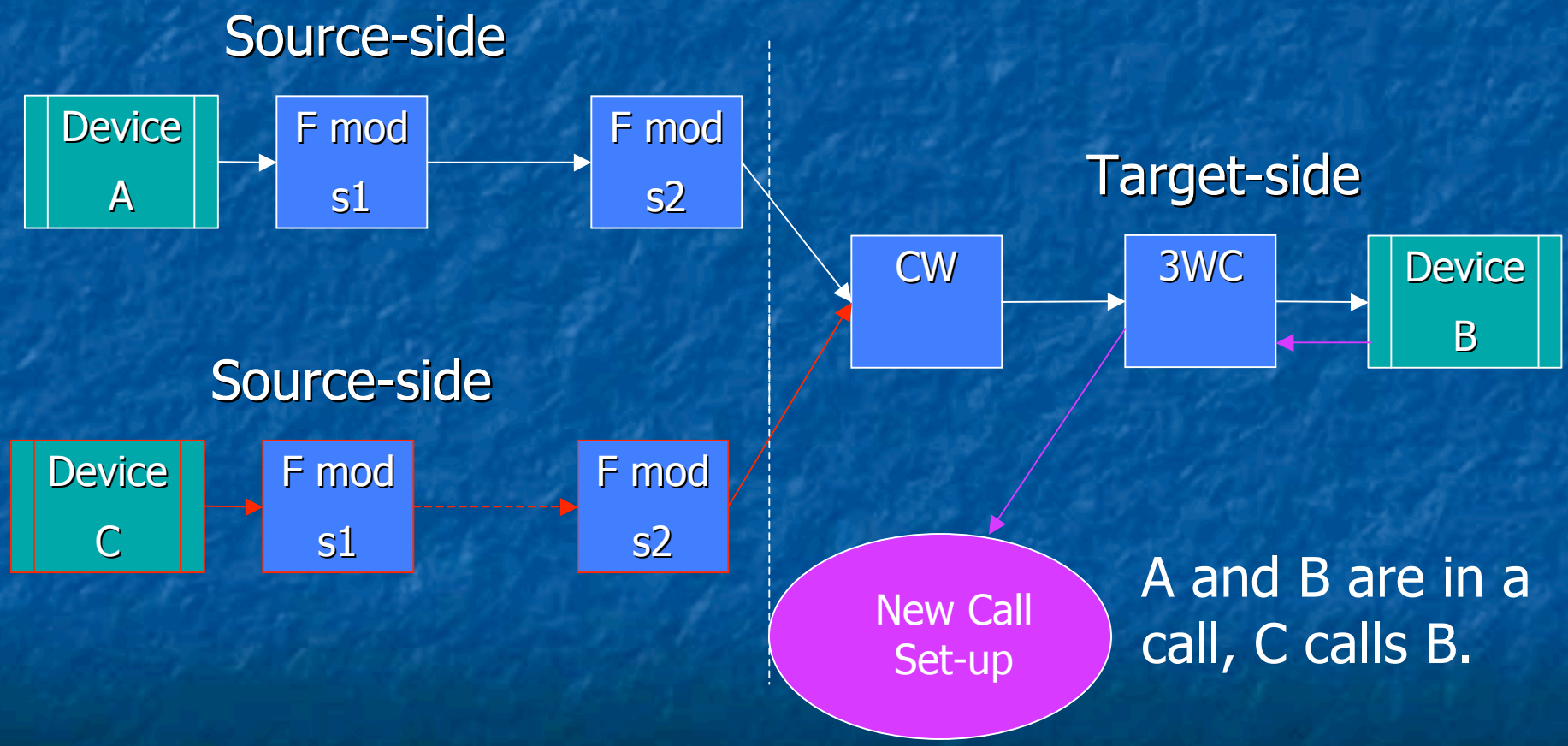


- Features are run in sequence

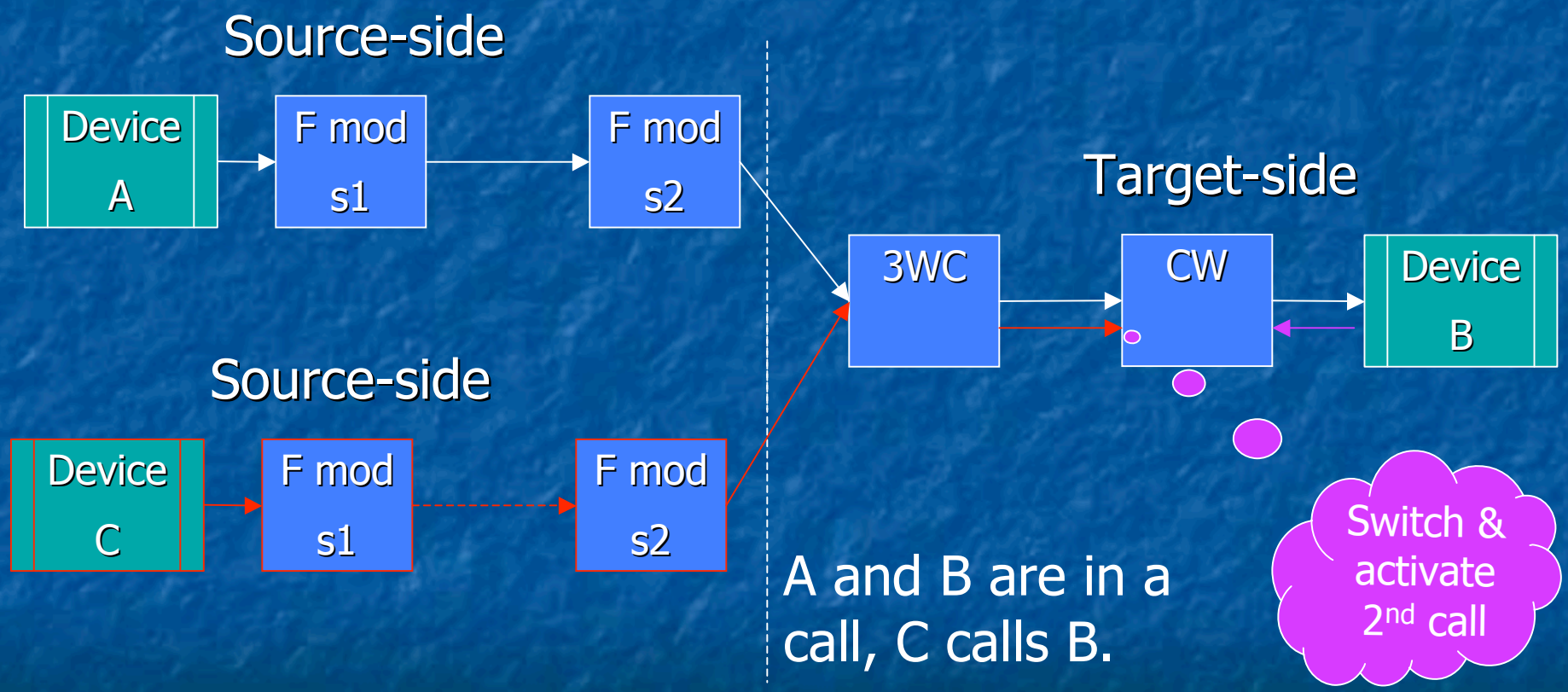
Feature Interaction Resolution

- Most feature interaction resolution is applied automatically, and is determined by the order of the feature modules in the call sequence.
- Hence, feature resolution in DFC mimics a priority resolution strategy.

Example: CW vs. 3WC



Example: CW vs. 3WC



DFC Advantages

- Non-linear calls are easy to set-up and handle.
- Model checking individual features and interactions between features is often possible because of the feature's design.
  - For example, SPIN has been used by Jackson and Zave to test for interactions in DFC.

DFC vs. COURT

- What are the difference between using a client-server design and a pipe and filter design for solving feature interactions?
  - Limited testing of possible resolution strategies in DFC.
  - COURT has more possibilities for computation and therefore is more complex and calculations may slow down connections.

DFC vs. COURT continued

- Coupling
  - In COURT coupling based on the states of the system.
  - In DFC coupling is based on shared reactions to signals.
- Separation of Concerns
  - Features in both COURT and DFC are written independently of each other.

Address Translation:
Aliasing, another problem in Telephony

- Address translation occurs when a feature changes the source or destination address of a call.
- Address translation causes many feature interactions (call forward vs. call forward).
- In DFC, Ideal address translation is done by adding additional constraints to the DFC architecture and feature modules that can prevent these types of interactions.

Conclusion

- Researchers are still exploring the many different ways software architecture can be used to resolve the feature interaction problem.