Deliberate Cooperation in Service-Oriented Environments

Mr David Paul, A/Prof Frans Henskens, Dr Michael Hannaford

Distributed Computing Research Group
School of Electrical Engineering and Computer Science
University of Newcastle, Australia

Cooperation

- Humans need to specialise
- People from different specialisations need to work together
- Cooperation is the process of working towards a common goal
    - Accidental cooperation
    - Forced cooperation
    - Deliberate cooperation

Service-Oriented Systems

- A provider offers a set of services
  - The provider's specialisation
- A consumer requests services
- Consumers often require services from multiple providers
  - Requiring providers to work together
    - Typically using accidental cooperation
- Using deliberate cooperation offers benefits for both consumers *and* providers

Motivating Example

Going to the theatre

- Ticket provider offers tickets to see the show
- Taxi provider offers transport to get to the show
- Consumer requires both a ticket and transport
    - If either is impossible, consumer wants neither
- With accidental cooperation, consumer is worse off if a provider fails
- Could force ticket provider to allow cancelling a booked ticket
    - May be unfair to provider
- Allow provider to choose whether to offer a cancelling ability
    - Leading to deliberate cooperation

Standard Levels of Cooperation

Example cooperation levels offered by current systems

- Full ACID support
  - Poorly suited to Web environment
- Semantic atomicity
  - Relaxes atomicity and isolation by allowing completed actions to be logically undone
- Tentative holds
  - Multiple consumers can have tentative holds on resources
  - When a consumer actually uses a resource, all other consumers with holds on that resource are informed their hold is no longer valid

Describing Different Levels of Cooperation

- Five basic operations:
  - Enquire (tentative holds)
  - Prepare (ACID)
  - Commit
  - Compensate (semantic atomicity)
  - Callback

- Resilience
  - Retry or try alternatives
  - Supported through abstract services

Supporting Dynamic Transactions

- Each provider indicates the level of cooperation it will support for each service call
  - May offer a different level for two subsequent requests
- Consumer reasons about cooperation offered by each provider in its workflow to ensure overall outcome is acceptable
  - Will only continue with workflow if willing to accept all possible outcomes

Choosing a level to support

Providers only offer levels of cooperation they are willing to support

- Each provider has a default cooperation level for a service
- Actual level changes based on provider considerations e.g.:
  - Currently available resources
  - Historical usage of the service
  - Historical usage by requesting client
  - Knowledge of competing services

Combining levels of cooperation

Consumers use the following algorithm:

- Prioritise activities based on need and risk
- If risk of performing next step of first activity is acceptable, perform next step
- Otherwise:
    - If risk is likely to be acceptable after waiting, continue
    - Else if alternative available, replace activity with alternative
    - Else if activity is optional, cancel the activity
    - Else cancel the workflow
- Repeat until workflow complete

Simulating Web Services Transactions

A simulator that models transaction flow rather than service flow

- Analytical results important to show correctness of possible solutions
  - Difficult to compare different possibilities
- Building actual systems expensive
  - Results only match the actual setup
- Simulation abstracts over unimportant details
  - Giving a low-cost addition to analysis that allows comparison of different solutions

Describing Scenarios

A scenario description contains details of:

- Concrete providers
  - Number of resources offered
  - Price
  - Likelihood of success
- Abstract providers
  - List of providers to use
- Client workflows
  - Activities to be performed
  - Whether activities are required or optional
- Timing information

Simulator Parameters

- Specified separately to scenario so differences can be directly compared
- Provider transaction support specified based on the five basic operations
    - Allows all reductions to be described when combined with abstract services
- Client's risk-taking behaviour specified based on budgets
    - High risk
        - Request all parallel actions immediately
    - Low risk
        - Request only undoable actions first

Simulator Operation

- Simulator is based on messaging model
  - Time modeller used to determine when messages are delivered
- Provider's state is tracked by simulator
  - Ensures correct transactional behaviour
  - For example, locked resources cannot be reused
- Client's interactions also monitored (on a per next action basis)
  - Depending on risk-taking behaviour, client will either:
    - Send message to begin next action
    - Cancel as much of the processed workflow as possible
    - Wait until the situation changes

Simulator Output

- As simulator is running, all messages are logged
  - Name of sender and receiver
  - Time message was sent and received
  - Content of message
- Allows extraction of information such as:
  - Length of a client's workflow
  - Whether a workflow completed successfully
  - Cost to the client
  - Number of a provider's resources that were utilised
  - Amount clients paid to a provider

Validation Experiment

Parameters

- Single provider offering 1000 resources
- Different levels of transaction support:
  - ACID prepare/commit scheme
  - ACID prepare/commit scheme with timeout of 500
  - Tentative hold Enquire/Callback/Commit scheme
  - Semantic atomicity Commit/Compensate scheme
- 1000 clients
  - Each booking between 1 and 10 resources
  - Other actions take up to 50 time units and fail 20% of the time

Validation Experiment

Assumptions

- Resource contention ensured
  - All transactions start between times 0 and 100
- Messages not received immediately
  - 1-5 time units required for message to be sent
- Processing is not instantaneous
  - Provider takes 1-10 time units to process a request

Validation Experiment

Results

- All provider's resources utilised except for when semantic atomicity is offered
- More transactions succeed when ACID properties enforced
  - Number of concurrently executing transactions reduced
- Transactions are significantly faster when reduced properties used

Transaction support	Provider utility (%)	Client successful (%)	Client unsuccessful without penalty (%)	Client unsuccessful with penalty (%)	Average duration of a transaction
ACID	100	23.0	77.0	0.0	1316
ACID with time out	100	24.5	72.8	2.7	1245
Tentative hold	100	15.6	57.5	26.9	228
Semantic atomicity	86.2	16.7	83.3	0.0	134

Verification Experiment

Description

- Three providers offering a competing service
  - Each consumer only uses one of these providers in their transaction (along with other actions)
- Service offers consumers a finite number of resources
  - Only difference between each provider is the cooperation level supported
- Each provider offers either:
  - Semantic atomicity – consumers can cancel without penalty
  - Tentative hold – consumers given non-exclusive reservation
  - Variable support – behaves as semantic atomicity when provider has plentiful resources, but switches to tentative hold when available resources drop below a threshold

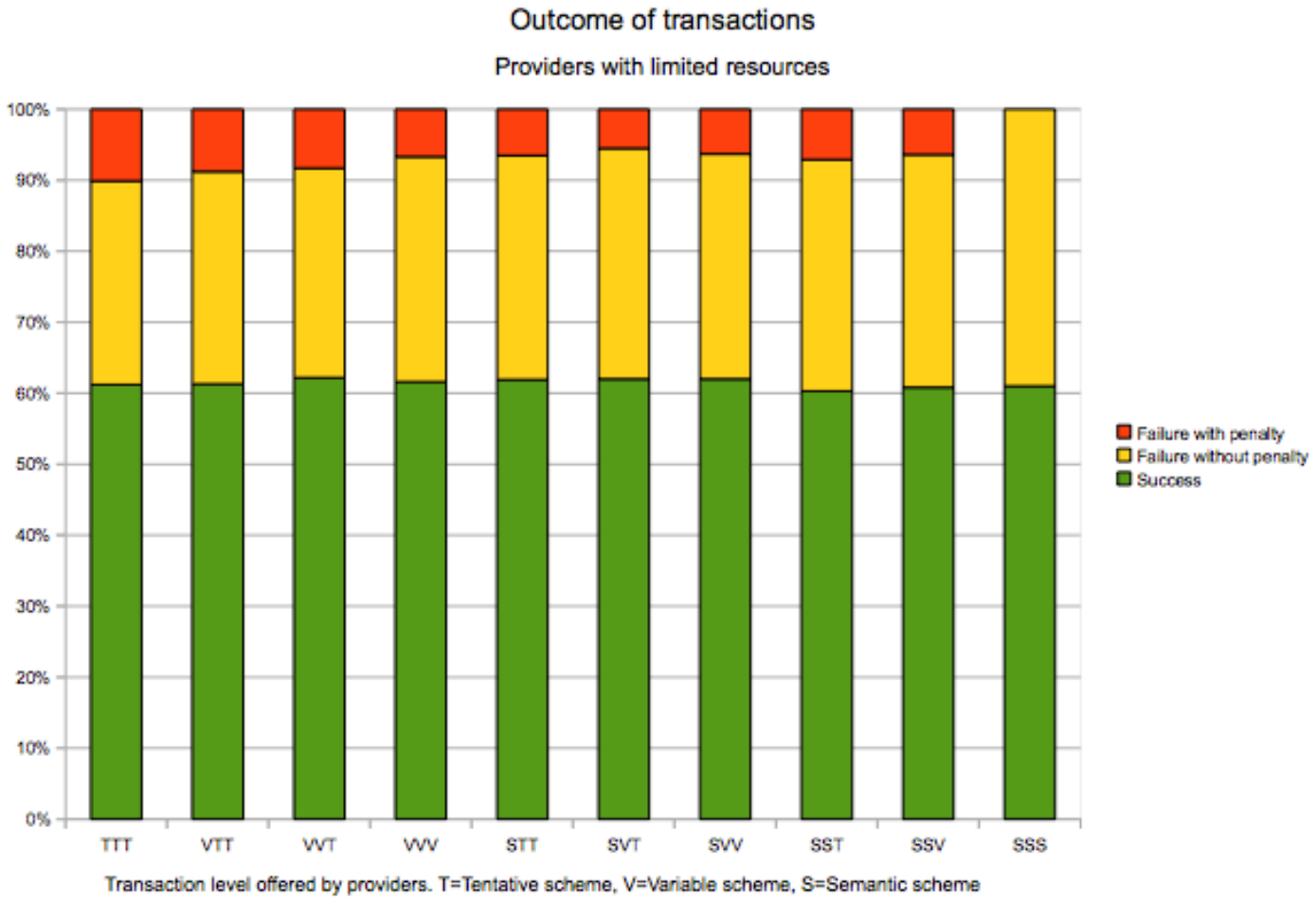
Verification Experiment

Parameters

- 1000 randomly-generated consumer transactions
  - 10% require semantic atomicity, 80% prefer semantic atomicity, 10% have no preference
- Simulation run with each provider offering each possible cooperation level
- Three experimental setups:
  - Each consumer requesting between 1 and 10 resources
    - Providers with limited resources
    - Providers with sufficient resources
  - Half of the consumers requesting 50 resources

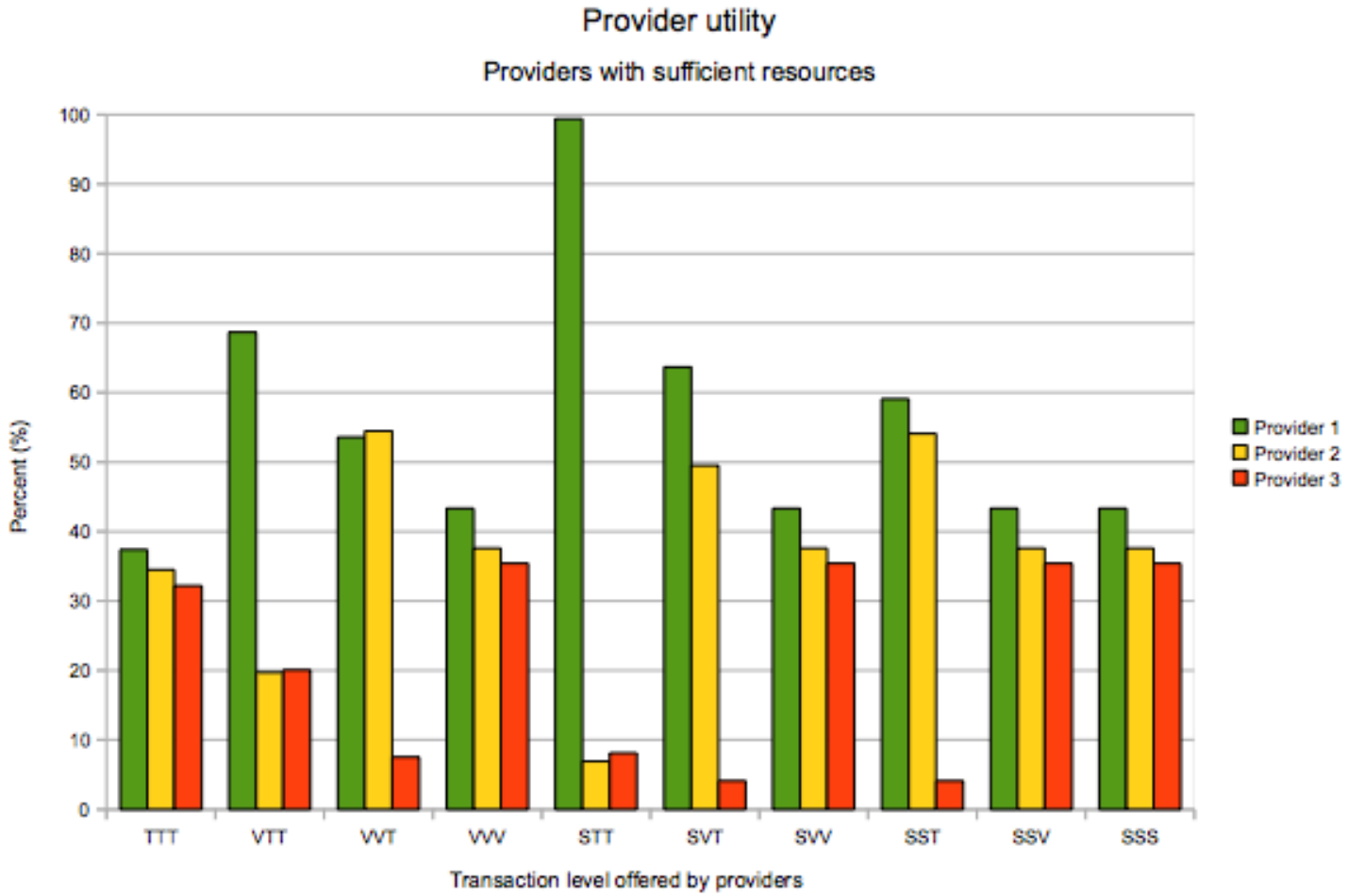
Verification Experiment

Results – Limited resources



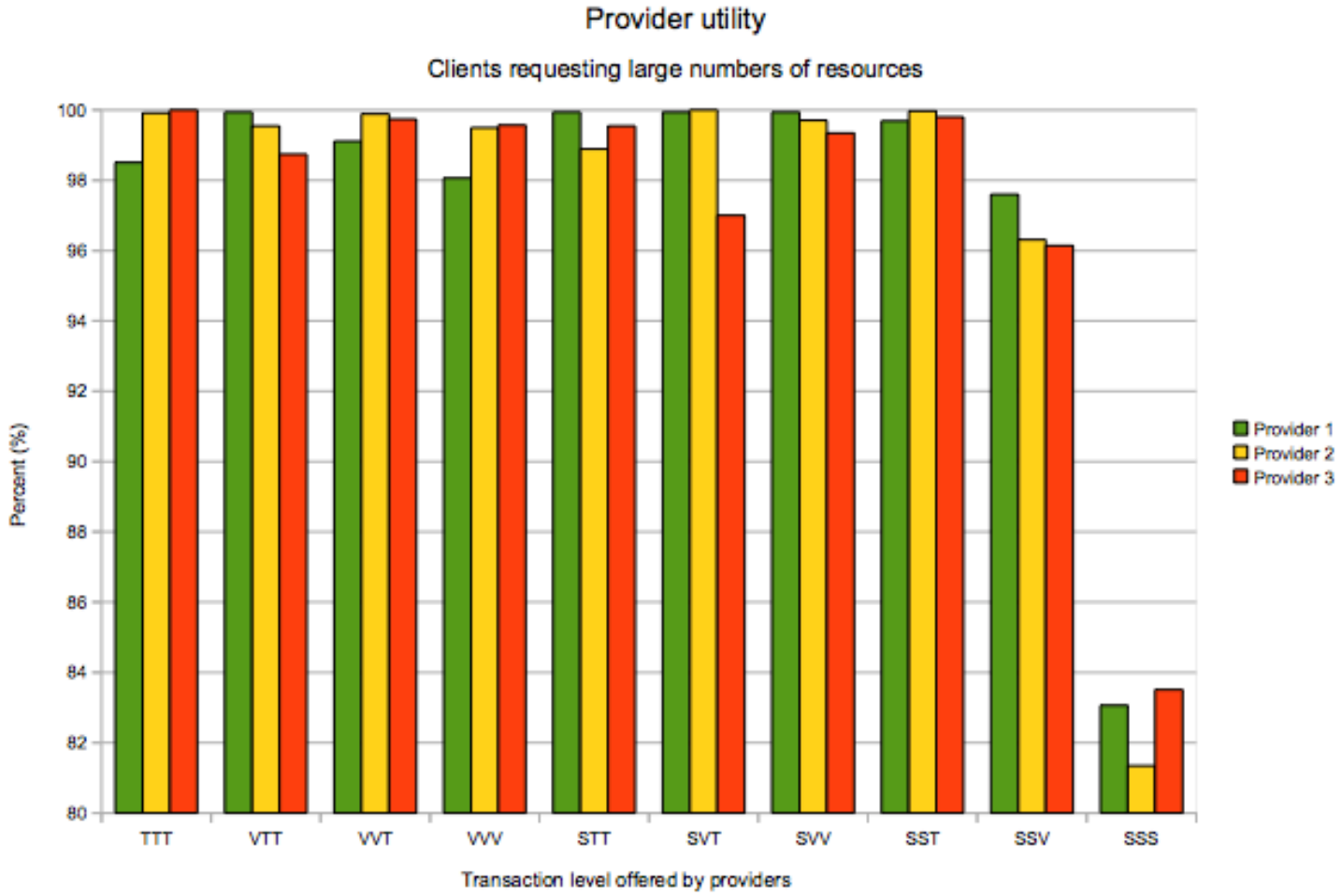
Verification Experiment

Results – Sufficient resources



Verification Experiment

Results – Clients requesting large amounts of resources



Conclusion

- Cooperation can be accidental, forced, or deliberate
- Cooperation can be thought of as level of transaction support
  - ACID often not the best choice in service-oriented environments
- Providers use deliberate cooperation when they can dynamically decide on the level of transaction support to offer for a service
- Consumers can combine services with different levels of support
- Simulation shows deliberate cooperation can be beneficial for both providers *and* consumers