A Contract Renegotiation Protocol

Michael Parkin
CoreGRID Research Fellow
Barcelona Supercomputing Centre

Overview

- Background
- Requirements
  - Message content, Protocol, Strategy
- Protocol Description & Specification
- Prototype Implementation

Background

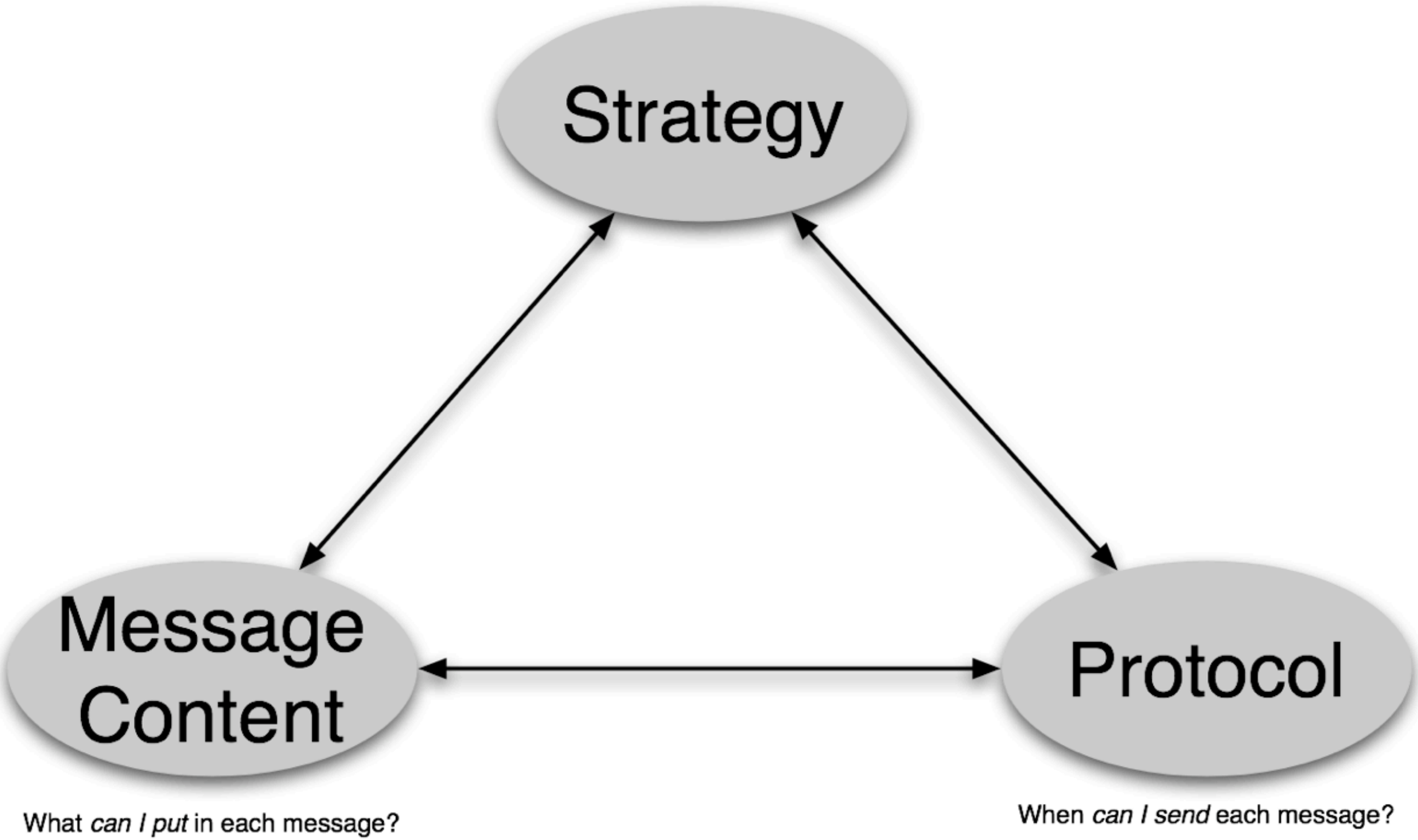
- Renegotiation protocol required for existing agreements
  - ...as circumstances of provider/consumer change
- Assume that we can't trust the other party to reply and an asynchronous, imperfect network
  - ... messages may be lost, duplicated and re-ordered
  - ... preferable to use a non-blocking protocol

Blocking

- Brewer's Conjecture: in a distributed system, there are three properties that are commonly desired:
  - consistency, availability, partition tolerance.
  - "...it is impossible to achieve all three"
  - In a grid, network partitioning is fundamental
  - Loosen consistency requirement to achieve available (i.e. non-blocked) resources
    - This protocol achieves consistency *eventually*

Requirements

When *do I want* to send each message?



What *can I put* in each message?

When *can I send* each message?

Message Content

- From the GRAAP-WG Wiki, some of the requirements for negotiation are related to message content. E.g:
  - "clearer information about why parties don't agree"
  - "renegotiable terms"
  - "reservation scenarios" (protocol should be scenario independent)
- Not covered here

Strategy

- Dependent on the operating/business model/ scheduling policy of the service provider and consumer
- Not covered here

Protocol

- This is covered here!
- Distinct roles: customer and resource provider
- Both parties can initiate multi-round renegotiation
- Initiation through non-binding enquiries (quote request and quote messages)
- Contract law offer-accept to form new contract. (Contract formed when accept is sent by the provider - c.f. ‘mailbox rule’)
  - No blocking as provider does not issue offers

Protocol States

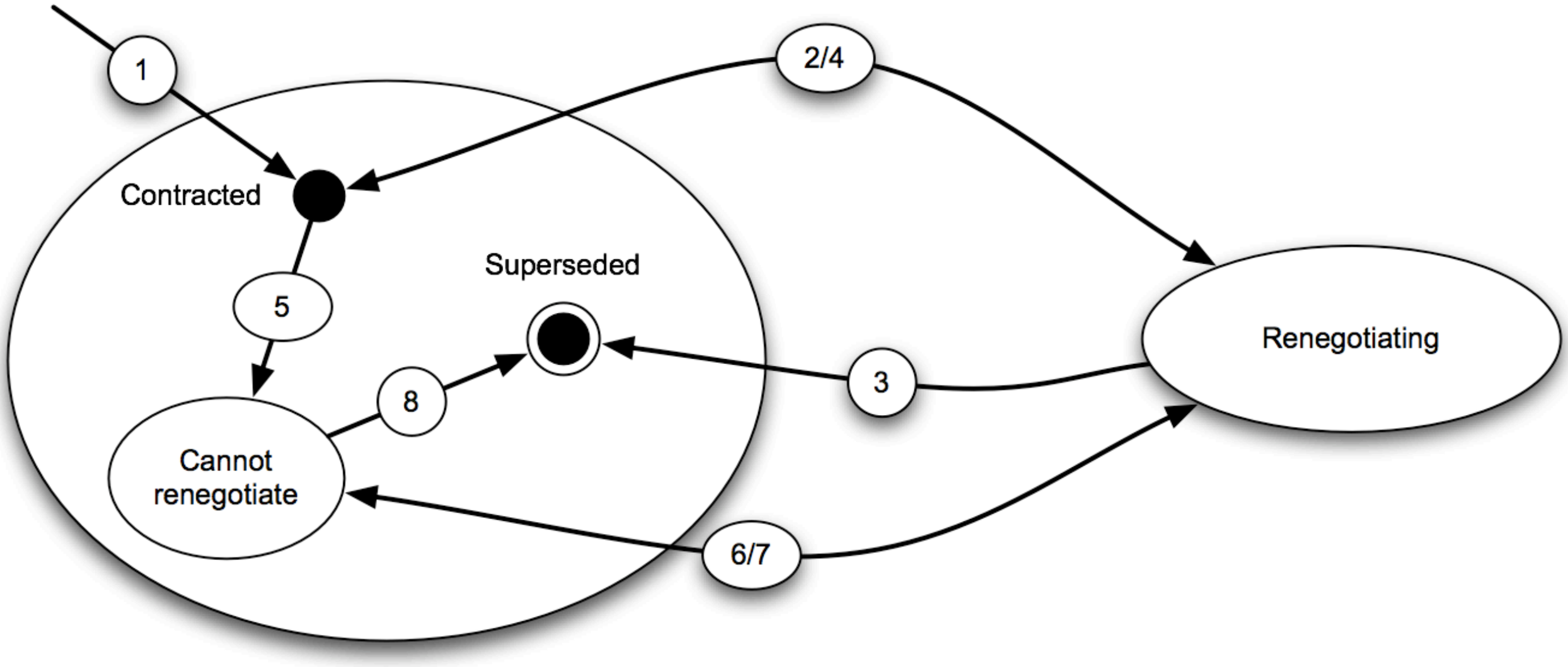
- Contract initially in the *Contracted* state
- Renegotiation initiated: *Renegotiating* state
- New, superseding contract agreed: *Superseded* state (a sub-state of contracted)
- If contract cannot be renegotiated*: *Cannot Renegotiate* state (a sub state of contracted)
  - *E.g. when a computational job is running

Protocol Messages

- RenegotiationQuoteRequest
- RenegotiationQuote
- RenegotiationOffer
- RenegotiationAccept
- RenengotiationReject
- CannotRenegotiate

Red: customer-provider
Black: provider-customer
Blue: both directions

Protocol State Machine



Implementation

- Written in Ruby-on-Rails
  - For integration with legacy Java/C/C++
- Interfaces/Representations
  - HTML (user-machine)
  - ROA/REST POX (machine-machine)
  - ATOM feed (notifications of state changes)
  - RPC/WSA not implemented (yet)

Implementation

- RESTful implementation allows us to have multiple contract representations
  - Plan to have a PDF representation for printing/ filing the contract
  - WS-Agreement representation for interoperation with ‘grid’ infrastructure
  - JSON representation for lightweight clients

Demo