# Tentamen Netcomputing 30 januari 2007

1. (a) Leg duidelijk uit wat de zogenaamde *middleware* laag is in gedistribueerde systemen, waar het in het lagenmodel past, en welke (nieuwe) functionaliteit het biedt.

   (b) Laat daarnaast zien wat in dit verband onder *transparancy* en *scalability* verstaan wordt en waarom het zo moeilijk te implementeren is.

2. (a) Leg duidelijk uit wat een *Remote Procedure Call* is en hoe het werkt. Laat daarbij met name zien hoe met de verschillen tussen de architecturen omgegaan wordt.

   (b) De OO-variant *Remote Method Invocation*, met name zoals in Java beschikbaar, is beter geschikt voor gebruik in gedistribueerde systemen. Leg het verschil met de RPC uit, en ook waar dit betere in zit.

3. (a) *Message-oriented middleware* zorgt voor asynchrone communicatie faciliteiten. Leg duidelijk uit welke extra communicatie mogelijkheden er ontstaan t.o.v. synchrone systemen zoals bijvoorbeeld RMI.

   (b) Geef een duidelijk voorbeeld, met enige uitleg, van een dienst die van deze extra mogelijkheden gebruik maakt.

4. (a) Voor het implementeren van gedistribueerde systemen kunnen we van veel verschillende *consistency modellen* gebruik maken. Wat is een consistency model, en wat kunnen we we met de verschillende modellen bereiken?

   (b) Wat wordt verstaan onder *causale consistentie*?

   Leg ook uit waarom de hieronder gegeven *store* wel causaal consistent, maar niet sequentieel- of strict- consistent is.

| | | | | |
|---|---|---|---|---|
| P1: W(x)a | | W(x)c | | |
| P2: | R(x)a | W(x)b | | |
| P3: | R(x)a | | R(x)c | R(x)b |
| P4: | R(x)a | | R(x)b | R(x)c |

5. Als er een fout in een gedistribueerd systeem optreed, bijvoorbeeld omdat een van de machine's crashed, moet er een mechanisme zijn om het systeem in een consistente toestand te brengen om vanaf daar weer verder te gaan.

   Leg duidelijk uit wat er in zo'n situatie te herstellen valt, en hoe dat zou kunnen gebeuren.