7 Concurrent and Distributed Systems (rmm1002)

(a) In the Network Time Protocol (NTP), a client (C) and a server (S) exchange (request, reply) messages to compute corrections to the time at C. Assume the time at S is always correct, and that C is synchronised to S at 13:30:00.

(i) Thirty days later the time at S is again 13:30:00 but C now believes the time to be 13:28:30. Define and compute skew and drift for C. [2 marks]

(ii) NTP estimates the offset and delay using four timestamps (T₀, T₁, T₂, T₃) from a request-reply message exchange. Two such exchanges occur between C and S, producing timestamps (310.000, 400.100, 400.102, 310.202) and (311.000, 401.150, 401.160, 311.410) respectively, denoting all timestamps as seconds since a common fixed point. Show on a diagram the point in the message exchange at which each timestamp T₀...T₃ is taken. Give definitions for offset and delay, and compute both for each set of timestamps. Which of the two offsets you have computed would you prefer to use to adjust the time at C, and why? [5 marks]

(iii) What happens to your estimates of offset and delay if network delays are no longer symmetric? [2 marks]

(b) It is often necessary to agree only on the ordering of events, not their times.

(i) x → y indicates event x happens-before event y. Define happens-before. Explain why it provides only a partial order on events. [2 marks]

(ii) Vector clocks can be used to implement happens-before. Give the vector clock values at each event, a...g, and explain whether each of the following relations is true or false: b → c, c → e, c → f, d → g. If false, give the relation that does hold between the given pair of events. [8 marks]

(iii) An earlier approach used Lamport Clocks, defining L(x) such that, for two events x and y, x → y ⇒ L(x) < L(y) but L(x) < L(y) ∉ x → y. Explain how vector clocks resolve this issue and ensure L(x) < L(y) ⇒ x → y. [1 mark]