Chapter 5

NETWORK CALCULUS

Roger Wattenhofer

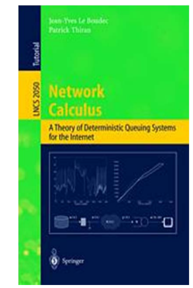
Overview

- Motivation / Introduction
- Preliminary concepts
- Min-Plus linear system theory
- The composition theorem

- Adversarial queuing theory
- Instability of FIFO
- Stability of LIS

- Sections 1.2, 1.3, 1.4.1
- Section 3.1
- Section 1.4.2

in Book "Network Calculus" by Le Boudec and Thiran



ica1www.epfl.ch/PS_files/NetCal.htm

What is Network Calculus/Adversarial Queuing Theory?

- Problem: Queuing theory (Markov/Jackson assumptions) too optimistic.
- Instead: Worst-case analysis (with bounded adversary) of queuing or flow systems arising in communication networks
- Network Calculus
  - Algebra developed by networking ("EE") researchers
- Adversarial Queuing Theory
  - Worst-case analysis developed by algorithms ("CS") researchers

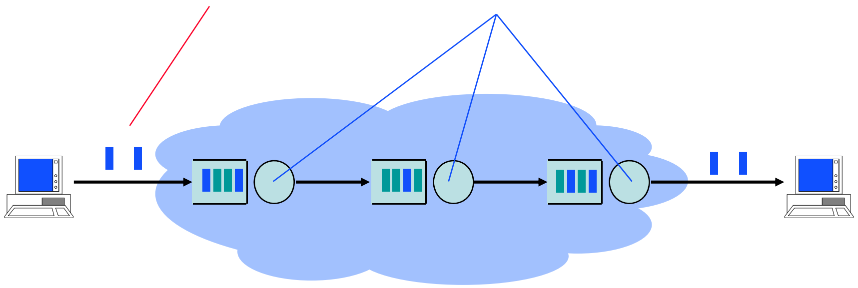
An example

- assume R(t) = sum of arrived traffic in [0, t] is known
- required **buffer** for a bit rate c is

  $\sup_{s \le t} \{R(t) - R(s) - c\cdot(t-s)\}$

Arrival and Service Curves

- Similarly to queuing theory, Internet integrated services use the concepts of *arrival curve* and *service curves*

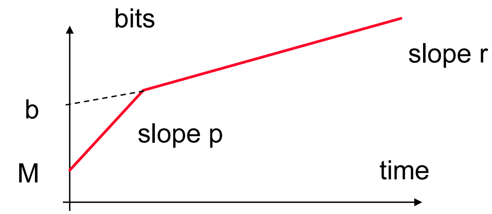


Arrival Curves

- Arrival curve α: $R(t) - R(s) \leq \alpha(t-s)$

Examples:

- leaky bucket $\alpha(u) = ru+b$
- reasonable arrival curve in the Internet
  $\alpha(u) = \min(pu + M, ru + b)$



Arrival Curves can be assumed sub-additive

- Theorem (without proof):

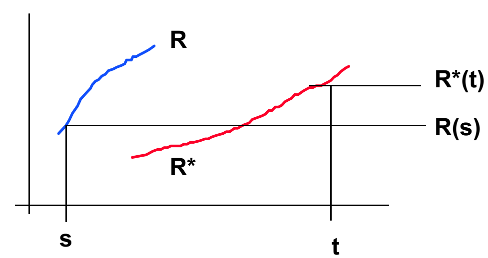
  α can be replaced by a *sub-additive* function

- sub-additive means: $\alpha(s+t) \leq \alpha(s) + \alpha(t)$
- concave $\Rightarrow$ subadditive

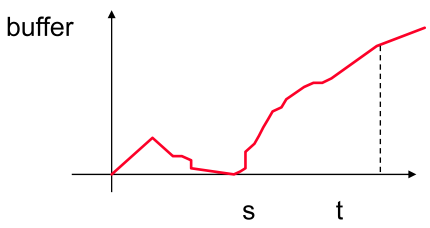
Service Curve

- System S offers a service curve β to a flow iff for all t there exists some s such that

$$R^*(t) - R(s) \geq \beta(t-s)$$



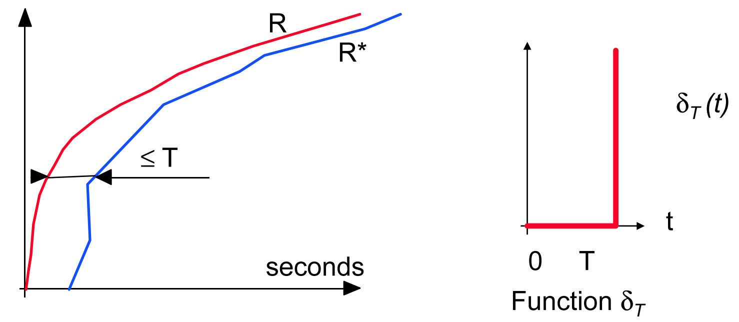
Theorem: The constant rate server has service curve $\beta(t)=ct$



Proof: take s = beginning of busy period. Then,

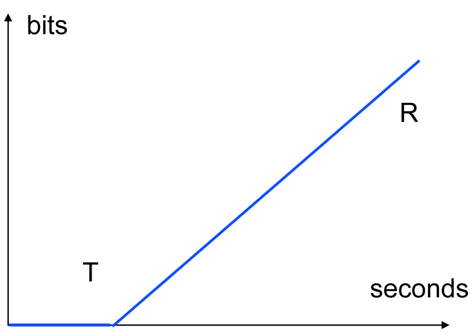
$$R^*(t) - R^*(s) = c \cdot (t-s)$$
$$R^*(t) - R(s) \geq c \cdot (t-s)$$

The guaranteed-delay node has service curve δ_T



A reasonable model for an Internet router

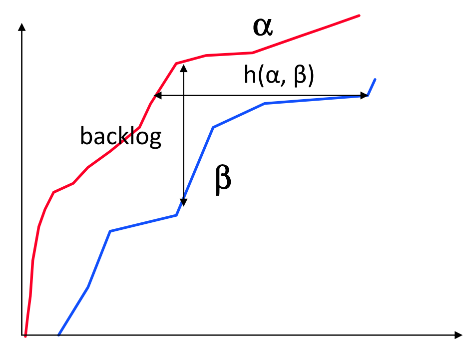
- rate-latency service curve



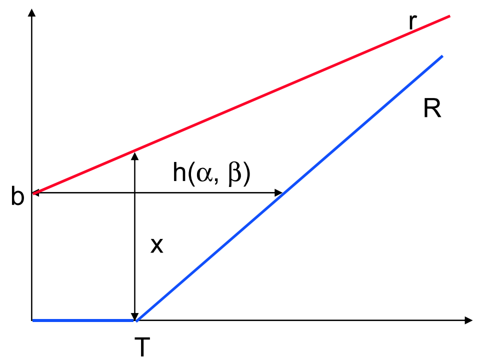
Tight Bounds on delay and backlog

If flow has arrival curve α and node offers service curve β then

- backlog $\leq \sup (\alpha(s) - \beta(s))$
- delay $\leq h(\alpha, \beta)$



For reasonable arrival and service curves



- delay bound: $b/R + T$
- backlog bound: $b + rT$

Another linear system theory: Min-Plus

- Standard algebra: R, +, ×

  $a \times (b + c) = (a \times b) + (a \times c)$

- Min-Plus algebra: R, min, +

  $a + (b \wedge c) = (a + b) \wedge (a + c)$

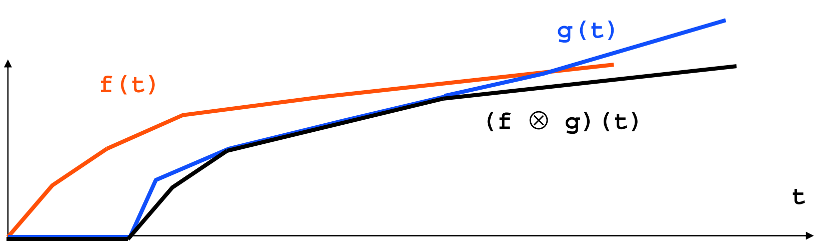
Min-plus convolution

- Standard convolution:

$$(f * g)(t) = \int f(t-u) g(u)\, du$$

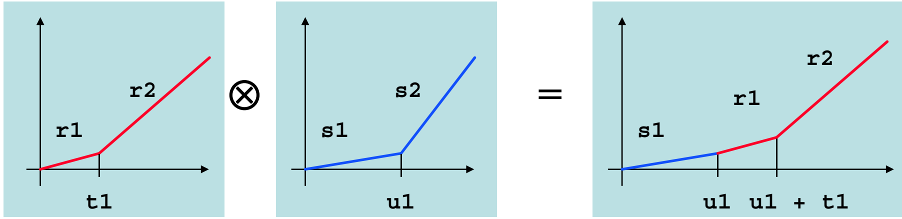
- Min-plus convolution

  $f \otimes g\ (t) = \inf_u \{ f(t-u) + g(u) \}$



Examples of Min-Plus convolution

- $f \otimes \delta_T\ (t) = f\ (t-T)$
- convex piecewise linear curves, put segments end to end with increasing slope



Arrival and Service Curves vs. Min-Plus

- We can express arrival and service curves with min-plus
- Arrival Curve property means

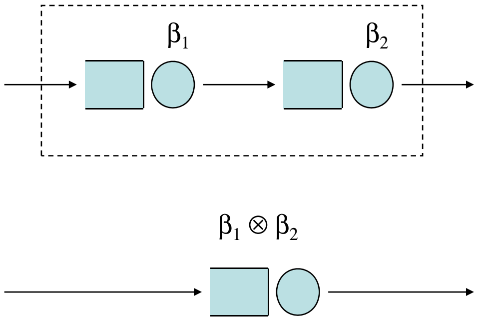
$$R \le R \otimes \alpha$$

- Service Curve guarantee means

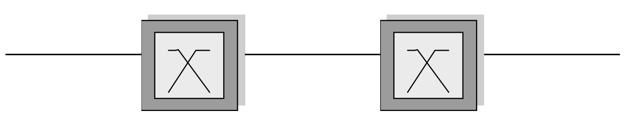
$$R^* \ge R \otimes \beta$$

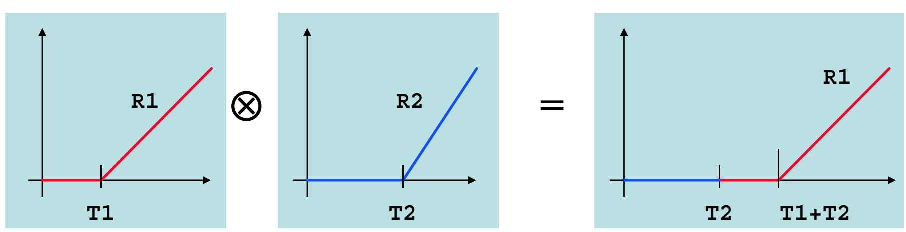
The composition theorem

- **Theorem**: the concatenation of two network elements offering service curves β_i and β_2 respectively, offers the service curve $\beta_1 \otimes \beta_2$

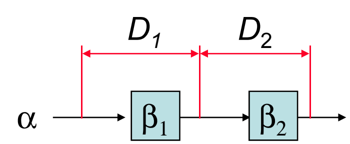


Example: Tandem of Routers

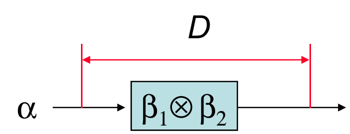




Pay Bursts Only Once



$$D_1 + D_2 \le (2b + RT_1) / R + T_1 + T_2$$



$$D \le b / R + T_1 + T_2$$

end to end delay bound is less

Adversarial Queuing Theory

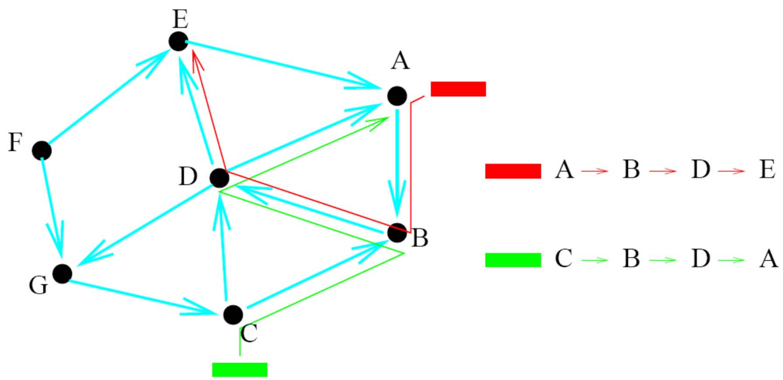
- We will revise several models of connectionless packet networks.
- We have a bounded adversary which defines the network traffic.
  - Like network calculus
- Our objective is to study stability under these adversaries.
  - If a network is stable, we study latency.
- [Thanks to Antonio Fernández for many of the following slides.]

Network Model

- The general network model assumed is as follows
  - A network is a directed graph.
  - Packets arrive continuously into the nodes of the network.
  - Link queues are not bounded.
  - A packet has to be routed from its source to its destination.
  - At each link packets must be scheduled: if there are several candidates to cross, one must be chosen by the scheduler.
- To make the analyses simpler initially, we assume
  - All packets have the same unit length.
  - All links have the same bandwidth.
  - This allows to consider a synchronous system, that is, the network evolves in steps. In each step each link can be crossed by at most one packet.

Example

- We are given two packets, each needs to cross three links.
- There is congestion on the link B→D, the execution needs 4 steps.



Adversarial Queuing Theory Model

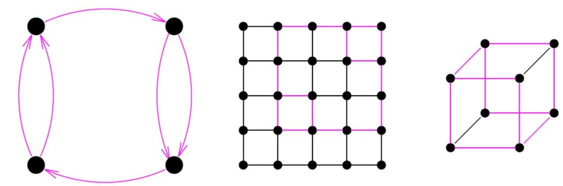
- [Borodin, Kleinberg, Raghavan, Sudan, Williamson, STOC96]
- [Andrews, Awerbuch, Fernandez, Kleinberg, Leighton, Liu, FOCS96]
- There is an adversary that chooses the arrival times and the routes of all the packets
- The adversary is bounded by parameters (r, b), where $b \geq 1$ is an integer and $r \leq 1$, such that, for any link e, for any $s \geq 1$, at most $rs + b$ packets injected in any s-step interval must cross edge e.
- We have a scheduling problem.

Stability

- A scheduling policy P is stable at rate (r, b) in a network G if there is a bound C(G, r, b) such that no (r, b)-adversary can force more than C(G,r,b) simultaneous packets in the network.
- A scheduling policy P is universally stable if it is stable at any rate $r < 1$ in any network.
- A network G is universally stable if it is stable at any rate $r < 1$ with any greedy scheduling policy.

Some Results

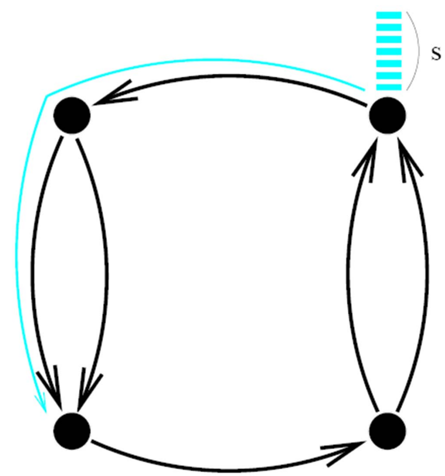
- Any acyclic directed graph (DAG) is universally stable, even for $r = 1$ [BKRSW01].
- The ring is universally stable
  - There are never more than $O(bn/(1 - r))$ packets in any queue.
  - A packet never spends more than $O(bn/(1 - r)^2)$ steps in the system.
  - Any added link makes the ring unstable with some greedy policy (for instance with Nearest-to-Go, NTG).
- FIFO is unstable for $r > 0.85$ with these networks:



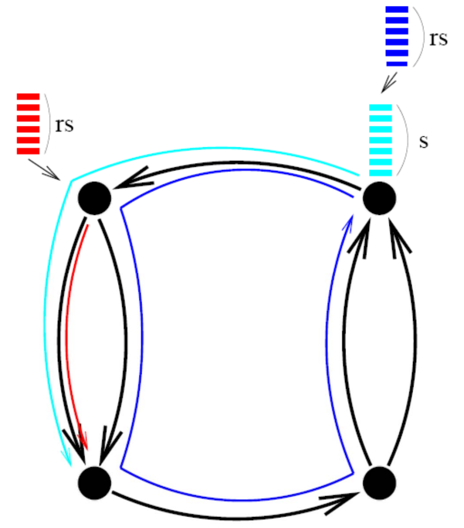
Proof of FIFO Instability

- Initially we have s packets in a queue with a given configuration.
  - Think of these packets to be inserted in an initial burst
- Then the algorithm proceeds in phases
  - Each phase is a bit longer than the phase before.
  - After each phase, we have the initial configuration, however, with more packets in a specific queue than in the previous phase.
  - By chaining infinite phases, any number of packets in the system can be reached.
- We show here the behavior of the adversary and the system in one phase.
  - Each phase has three rounds.

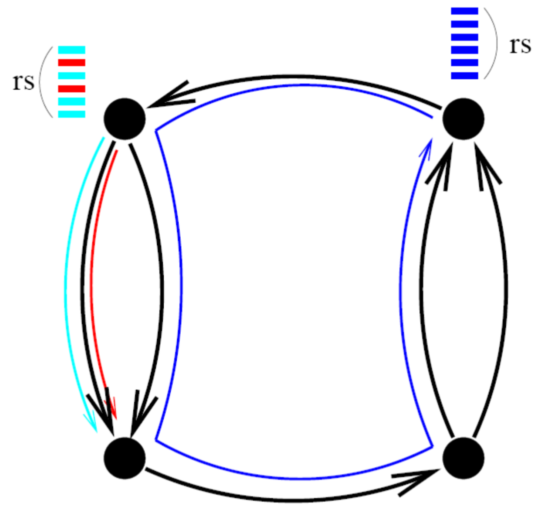
Initial Situation



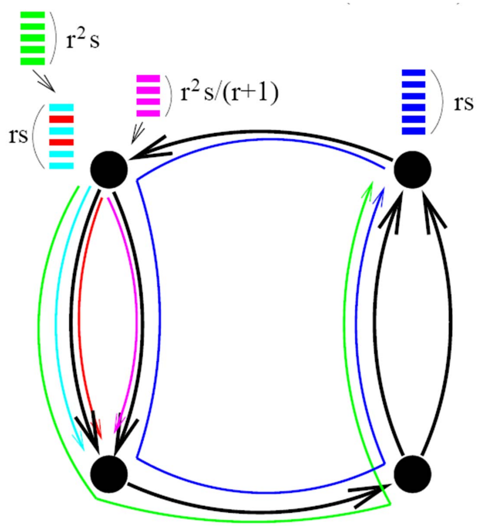
Injecting packets in the first round (s steps)



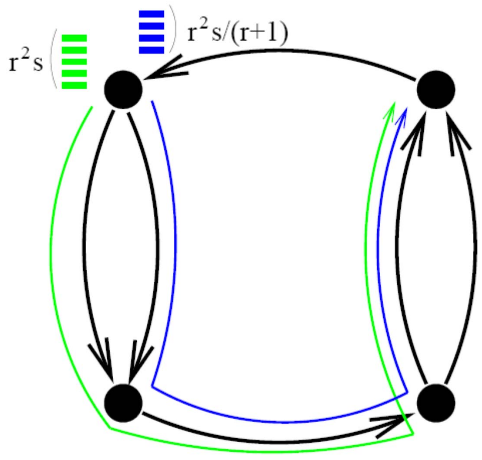
Situation after the first round



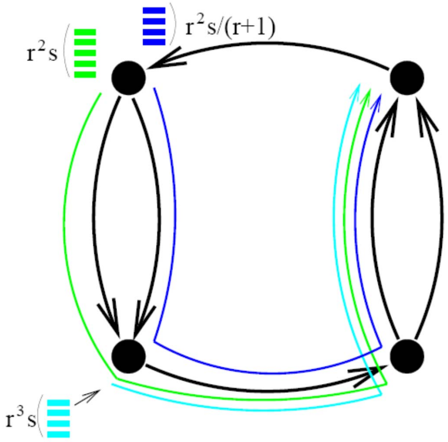
Injecting packets in the second round (rs steps)



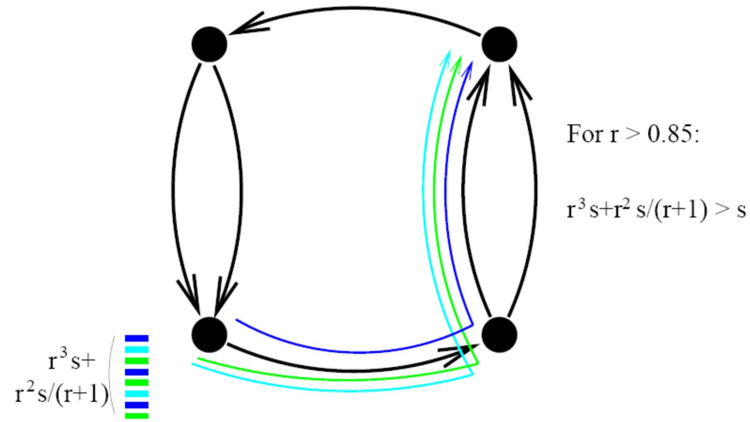
Situation after the second round



Injecting packets in the third round (r^2s steps)



Final situation (end of phase, after the third round)



More Results

- Several simple greedy policies are universally stable
  - Longest-in-System (LIS): Gives priority to oldest packet (in the system).
  - Shortest-in System (SIS): Gives priority to newest packet (in the system).
  - Farthest-to-Go (FTG): Gives priority to the packet farthest from destination.
  - Nearest-to-Source (NTS): Gives priority to the packet closest to its origin.
- All mentioned greedy policies can suffer delays that are exponential in d, where d is the maximum routing distance.
  - Moreover, any deterministic policy that does not use information about the packet routes to schedule can suffer delays exponential in √d [Andrews Z 04].
  - There are deterministic distributed algorithms that guarantee polynomial delays and queue lengths [Andrews FGZ 05].

Universal stability of LIS (Longest-in-System)

- Network G, adversary in bucket AQT with parameters $r = 1-\varepsilon < 1$ and $b \geq 1$.
- Def.: Class L is the set of packets injected in step L.
- Def.: A class L is active at the end of step t if there are some packets of class $L' \leq L$ in the system at the end of step t.
- Let us consider a packet p injected in step T_0. Packet p must cross d links, it crosses the i-th link in step T_i.
- Def.: c(t) is the number of active classes at the end of step t. Let $c = \max_{T_0 \leq t < T_d} c(t)$, that is the maximum number of active classes during the lifetime of packet p.

Lemma: $T_d - T_0 \le (1-\varepsilon^d)(c + \frac{b}{1-\varepsilon})$.

- p arrives to the queue of its i^{th} link in T_{i-1}.
- Only the packets in c – (T_{i-1}–T_0) active classes can block p.
- There are no more than (1-ε)(c+T_0–T_{i-1}) + b packets in these classes (p included), that is at most (1-ε)(c+T_0–T_{i-1}) + b–1 packets can block p. Then,

$$\begin{aligned} T_i &\le T_{i-1} + (1-\varepsilon)(c + T_0 - T_{i-1}) + b \\ &= \varepsilon T_{i-1} + (1-\varepsilon)(c + T_0) + b. \end{aligned}$$

$$\begin{aligned} T_d &\le ((1-\varepsilon)(c+T_0)+b)\sum_{i=0}^{d-1}\varepsilon^i + \varepsilon^d T_0 \\ &= ((1-\varepsilon)(c+T_0)+b)\frac{1-\varepsilon^d}{1-\varepsilon} + \varepsilon^d T_0 \\ &= (1-\varepsilon^d)(c + \frac{b}{1-\varepsilon}) + T_0 \end{aligned}$$

Lemma: Bounding both classes and steps

- Let t be the first time when either the system features more than c classes, or there is a packet in the system for more than c steps, for some c.
- Clearly, “classes” cannot be violated first, because there can only be c+1 classes if there is at least one packet in the system for at least c+1 steps.
- So we know that “steps” must be violated first. Let p be a first packet which is in the system for at least c+1 steps. (Note that during this time, we had at most c classes.)
- Let c = b/((1-ε)ε^d). Then the packet p cannot be in the system for more than c steps, because using our previous lemma (and b≥1 and ε>0), the number of steps of p is bounded:

$$(1-\varepsilon^d)(c + \frac{b}{1-\varepsilon}) + 1 = c - \varepsilon^d b/(1-\varepsilon) + 1 < c + 1$$

Theorem: LIS is universally stable

- Each packet leaves the system after c = b/((1-ε)ε^d) steps.
- In addition one can show that there are at most b+b/ε^d packets in each queue at all times.
- That’s all folks!