Modelos y Algoritmos para lograr QoS en Redes Industriales

Danilo Ávila Cárcamo 201321032-5

I. Introducción

Debido a los avances tecnológicos y para así facilitar sus operaciones es que las empresas han buscado integrar el uso de redes industriales en los procesos que llevan a cabo. Esto no ha sido tarea sencilla debido a que a diferencia de las redes usadas a diario, las redes industriales deben tener asegurada una calidad de servicio (QoS) para que no existan errores en los procesos que realizan.

En el presente trabajo se estudiara un modelo y algoritmos los cuales permitan lograr la buscada QoS, para esto se usará un modelo basado en redes definidas por software (SDN), a su vez se estudiaran algoritmos los cuales sean capaces de permitir nuevo flujo y asignar recursos en la red con el fin de lograr entregar QoS.

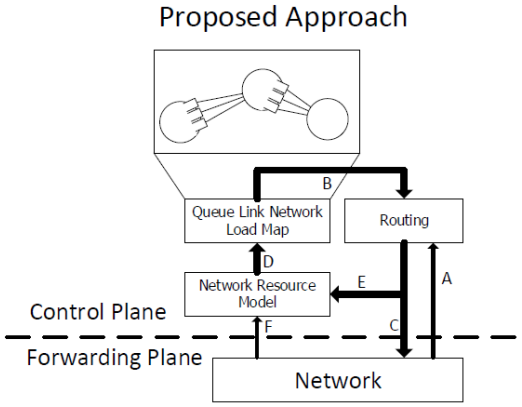

Fig. 1: Modelo Propuesto

II. Problema general

A. QoS y redes de computadores

Las redes industriales transmiten datos críticos, tales como señales de control para distintas instalaciones, estos mensajes deben ser enviados con restricciones en el tiempo de entrega, sin errores y sin perdidasa, el problema principal básica en como modelar y asignar los recursos a la red con el fin de obtener QoS. Antes de la aparición de las SDN, distintos tipos de arquitecturas buscaron lograr entregar QoS, algunas de estas son: PROFINET, EtherCAT, EtherNET/IP, SERCOS III etc, las cuales no han sido adoptadas de manera masiva debido al no ser muy eficientes y a su costo. Sin embargo con la llegada de las Redes Definidas por Software (SDN) se logró un método que permite de manera flexible y sencilla controlar redes de computadores, esta capacidad de controlar la red mediante aplicaciones es lo que le da la capacidad de poder implementar de manera eficiente QoS.

En el presente trabajo se estudiará principalmente algoritmos de admisión de flujo y asignación de recursos con el fin de obtener hard real-time industrial QoS. Para esto es que primero se define un modelo de red (figura 2), llamada de aqui en adelante ”queue link”. La idea de este modelo sera que los recursos van a ser activamente asignados a una de las llamadas colas o dependiendo de los requerimientos mínimos QoS de servicio que se debe lograr . Luego un algoritmo de ruteo, por ejemplo un sistema de ruteo de bajo-retardo y mínimo-costo (DCLC routing) [4], puede encontrar rutas a través de la SDN que cumplan con los requerimientos de la QoS.

III. Logrando comunicación en tiempo real usando SDN

A. ¿Cómo abordad el problema?

La comunicación en redes industriales debe cumplir requerimientos en tiempo real con retardos permitidos muy pequeños, conocido como requerimientos del tipo Hard Real-Time, en este caso si la transmisión de datos no se completa en un tiempo determinado la conexión se debe considerar como terminada. A su vez existen los requerimientos del tipo soft Real-Time en los cuales las conexiones se consideraran perdidas si el plazo para entregar los paquetes no se cumple varias veces. En el presente trabajo nos centraremos en el primer tipo de requerimiento.

Las SDN nos permiten tener acceso al llamado plano de datos, lo que viene siendo la parte del router que decide que hacer o donde enviar un paquete entrante a este, es fácil darse cuenta que esto será de gran utilidad a la hora de querer lograr QoS, para esto se tendrá que armar un modelo (SDN) de la red donde se tendra el estado y recursos físicos de esta, lo cuales seran, el tamaño del buffer, el número de enlaces en cada switch o router, la velocidad de transmisión de cada enlace y la topología de la red. Por otra parte las aplicaciones se limitaran a demandar cierta QoS, maximo retardo, máximas perdidas aceptables, jitter, etc. Finalmente el controlador de la SDN ,basado en su conocimiento sobre la red creada debe se capaz de decidir como lograr los requerimientos pedidos por la aplicación.

B. Modelo Usado

Para llevar a cabo el ruteo y asignación de recursos es que se tendra, en forma general, el modelo que se ve en la Figura 1 en donde en A se recibiran los requerimientos desde la red al controlador, en B lo que se hace es que desde un modelo virtual de la red cargado en el controlador se buscará el mejor camino posible para los datos a enviar, esto basado en último estado de la red, en C se entrega este camino a la red y los datos a enviar, una vez se esta con un nuevo flujo de datos se debe volver a actualizar el modelo virtual de la red (D), para esto es que se debe mantener actualizado un modelo de recursos en la red (Network Resource Model) el cual se actualiza mediante datos estáticos de la red (F), tales como topología, velocidad de enlace físico y capacidad de buffer, y también dependiendo de como estan las conexiones actualmente en la red (E).

El modelo que se usará, llamado "Queue-Link Network Topology" se puede ver en la figura 2

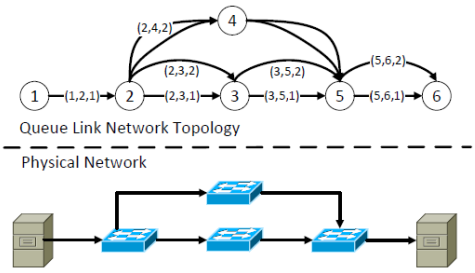

Fig. 2: Ejemplo ilustrativo de red con topologia queue link, en este ejemplo se tiene un nodo de origen, uno de destino y 4 nodos de switcheo, cada nodo tiene 2 enlaces $Q_{u,v} = 2$. Un enlace que en este caso también se llamaran Edge (borde o arista) (u, v, q) conecta la cola de prioridad q desde el nodo u al v.

C. Ruteo y admisión de control

Se tendra una red $G = (N, E)$ compuesta de N nodos y distintos edges o enlaces E, conociendo el tiempo máximo de retardo permitido desde inicio a fin, t_f, y siendo el origen un nodo "o" y destino un nodo "d" se tendran distintas rutas $P(o, d)$, donde el delay en cada ruta ($D(P(o, d))$ será menor a t_f por lo tanto seran rutas admitidas. Luego usando algun algoritmo de enrutamiento como por ejemplo uno de DCLC [4], el cual en forma general usa un modelo donde se le asigna un costo dinstinto a cada ruta desde el origen a destino, se busca el recorrido con el menor costo y luego se puede resolver en el controlador SDN el problema que busca minimizar la función:

$$C(P_i) \tag{1}$$

Donde $C(P_i)$ se conoce como el costo de tomar el camino P_i.

Para lograr la admisión de flujo será importante conocer el delay que existe en cada cola, se tomará el peor caso de retardo

TABLE I: Parámetros

N	Modelo de la Red
$Q_{u,v}$	Nodos en la red.
(u, v, q)	Número de colas de prioridad en enlace (u, v).
$R_{u,v}$	Conexión desde nodo u a v por cola q.
$E = (u, v, q)$	Bit rate de enlace (u, v).
$G = (N, E)$	Edges, o aristas en la red.
	Modelo completo de la red.
r_f	Modelo del flujo de datos
b_f	Bit rate promedio de un flujo f.
t_f	Burst o máxima ráfaga de bytes del flujo f.
	Delay máximo permitido para el flujo f desde origen a destino en la red.
$A_{R[u,v,q]}$	Modelo de recursos en la red
$A_{B[u,v,q]}$	Bit rate asignado a (u, v, q).
$U_{R[u,v,q]}$	Capacidad del buffer asignada a (u, v, q).
$U_{B[u,v,q]}$	Bit rate usado en (u, v, q)
$I_{R[n,u,v,q]}$	Capacidad del buffer usado en (u, v, q)
$I_{B[n,u,v,q]}$	Bit rate desde nodo n a (u, v, q)
	Máxima ráfaga desde nodo n a (u, v, q)
$S(u, v, q)$	Bit rate disponible para (u, v, q)
$T[u, v, q]$	Retardo máximo para $[u, v, q]$
$P(o, d)$	Posibles caminos desde un origen o a destinto d.
P_f	Ruta generada mediante DCLC
$C(P_i)$	Costo de usar camino P_i
$D(P_i)$	Peor delay desde origen a destino.

en cada cola $T[u, v, q]$,esto nos permitira escribir el máximo retardo posible de la siguiente manera:

$$D(P_i) = \sum T[u, v, q] \tag{2}$$

Cada flujo de información nueva a entrar a la red debe cumplir las siguientes restricciones:

- 1) La nueva tasa de bits entrantes (r_f) más la tasa de bits utilizada actualmente en un enlace no puede superar la tasa de bits máxima de este, esto quiere decir $U_R[u, v, q] + r_f \leq A_R[u, v, q]$.
- 2) La utilización del buffer en un enlace no puede superar al maximó buffer asignado para este $U_B[u, v, q] + b_f \leq A_B[u, v, q]$.
- 3) Finalmente el delay visto en la ecuación (2) tiene que ser menor al delay permitido del flujo entrante $D(P_i) \leq t_f$.

Estas 3 restricciones seran la base para el control de admisión de nuevo flujo, acontinuación se procederá a mostrar y explicar el algoritmo de ruteo usado.

- Lineas 2-3, se buscan todas las colas (queues) que cumplan con la restricción 1) y 2) sobre la utilización máxima de bit rate y buffer.
- Linea 4, se generá un gráfico de la red a partir de los edges y nodos obtenidos.
- Linea 5, se creará un camino P_f generado por un algoritmo DCLC [4] el cual considerará la restricción de retardo 3) para generar las distintas rutas, en caso de que no exista tal camino se termina la conexión y en caso contrario se procederan a actualizar los valores de I_R, I_B, U_R, U_B.
- Linea 9-17, para cada enlace utilizado se procedera a actualizar, utilizando la tasa de bits entrantes I_R y U_R y usando el máxima rafaga de bits entrantes I_B, la utilización de buffer se considerará como si estuviera

```
1: procedure ADD-FLOW(f)
2:     E*(f) = {(u,v,q) ∈ E | A_R[u,v,q] > U_R[u,v,q] + r_f
3:                           ∧ A_B[u,v,q] > U_B[u,v,q] + b_f}
4:     G*(f) ← (N, E*(f))
5:     P_f ← DCLC-Algorithm(G*(f), t_f)
6:     if P_f == null then
7:         return False
8:     end if
9:     for all p = 1, 2, ..., p_max,f do
10:        Upd. incom. rate, burst from prior node n_p = u_{p-1}:
11:        I_R[n_p, u_p, v_p, q_p] ← I_R[n_p, u_p, v_p, q_p] + r_f
12:        I_B[n_p, u_p, v_p, q_p] ← I_B[n_p, v_p, u_p, q_p] + b_f
13:    end for
14:    for all p = 1, 2, ..., p_max,f do
15:        Upd. rate, buffer util. at switch node u_p:
16:        U_R[u_p, v_p, q_p] ← U_R[u_p, v_p, q_p] + r_f
17:        U_B[u_p, v_p, q_p] ← B_max(u_p, v_p, q_p)
18:    end for
19:    return True
20: end procedure
```

Fig. 3: Algoritmo para admisión de flujo

utilizado al máximo para que no ocurran problemas por rafagas de datos entrantes.

Una vez finalizada una conexión se deben volver a actualizar I_R, I_B, U_R, U_B, solo que en este caso en vez de sumar r_f y b_f estos se restan.

D. Asignación de recursos Offline

Con el modelo de red en el trasfondo periodicamente se estará ejecutando la asignación de recurso a cada "queue link". La idea es optimizar la asignación de recursos de cada enlace físico a sus correspondientes queue links, se busca eliminar los llamados "cuello de botellea", lo que vendria siendo que mucha información pase por enlaces que no tienen la capcidad necesaria. Teniendo en cuenta una estimación de cada cuanto tiempo entrara nuevo flujo a la red, se puede hacer que la asignación offline de recursos se produzca periódicamente.

A continuación se presentará un algoritmo para la asignación de recursos:

Inicialmente la asignación de recursos a la red se hará acorde a predicciones sobre redes similares a la que se esta utilizando, esto no tiene demasiada importancia ya que luego de esto periodicamente se hará la asignación de recursos siguiendo el algoritmo que se ve en la figura 4.

Primero se copiara el estado actual de la red a una red virtual, esto se ejecuta en la linea 4 del código. Desde la línea 11 se realizará la asignación de tasa de bits a cada "queue link" ($A_R[\sigma, \omega, q]$) dependiendo de la capacidad máxima (R) del enlace físico que existe entre un nodo (σ, ω), por ejemplo si se tiene un enlace físico de 1[Gb/s], se pueden asignar valores tales que $\sum(A_R[\sigma, \omega, q]) \leq 1[Gb/s]$.

Para una tasa de bits asignada ($A_R[\sigma, \omega, q]$) con q pertenecience solo a "queue links" entre (σ, ω), se puede estimar el flujo máximo admitido con el algoritmo de la figura 5.

Donden entre las líneas 2 y 6 los flujos que estan actualmente circulando por la red son agregados a la red virtual

```
1: procedure RESOURCEALLOCATION
2:     for all Switching Nodes σ ∈ N do
3:         for all Next-hop Nodes ω ∈ O_σ do
4:             Virtual Netw. Res. Model ← Netw. Res. Model
5:             Execute Steps on Virtual Netw. Res. Model
6:             T_{σ,ω} = Set of admitted flows traversing σ, ω
7:             for all f ∈ T_{σ,ω} do
8:                 REM-FLOW(f)
9:             end for
10:            F_max = 0
11:            for all A_R[σ, ω, q]_{q ∈ L_{σ,ω}} ∈ R do
12:                F = FlowCountEst(A_R[σ, ω, q]_{q ∈ L_{σ,ω}})
13:                if F > F_max then
14:                    F_max = F
15:                    A_{R,max} = A_R[σ, ω, q]_{q ∈ L_{σ,ω}}
16:                end if
17:            end for
18:            if F_max > |T_{σ,ω}| then
19:                Netw. Res. Model ← A_{R,max}[σ, ω, q]_{q ∈ L_{σ,ω}}
20:                for all (σ, ω, q) ∈ L_{σ,ω} do
21:                    Evaluate T[σ, ω, q] from Eqn. (12)
22:                end for
23:                for all f ∈ T_{σ,ω} do
24:                    REM-FLOW(f), ADD-FLOW(f)
25:                end for
26:            end if
27:        end for
28:    end for
29: end procedure
```

Fig. 4: Algoritmo para asignación de recursos

```
1: procedure FLOWCOUNTEST(A_R[σ, ω, q]_{q ∈ L_{σ,ω}})
2:     for all Admitted flows f ∈ T_{σ,ω} do
3:         if ADD-FLOW(f) = False then
4:             return 0
5:         end if
6:     end for
7:     FlowCount = |T_{σ,ω}|
8:     New flow n = StatModel(T_{σ,ω})
9:     while ADD-FLOW(n) = True do
10:        New flow n = StatModel(T_{σ,ω})
11:        FlowCount++
12:    end while
13:    return FlowCount
14: end procedure
```

Fig. 5: Algoritmo para estimación flujo máximo

generado en la linea 4 de la figura 4, luego de esto entre las lineas 8 y 12 se estima la cantidad de flujo extra que podrian atravesar el enlace (σ, ω), esto se puede hacer usando algún modelo estadístico con los parámetros del flujo que esta atravesando actualmente las distintas colas. La cantidad de flujo actualmente en el enlace más la cantidad estimada de flujo que puede ser admitida es entregada al algoritmo de asignación de recursos de la figura 4.

El nuevo $(A_R[\sigma, \omega, q])$ a asignar se busca de tal forma que este permita obviamente que el flujo que circula por el enlace pueda seguir pasando por el sin problemas al actualizar los recursos de la red y que además el flujo estimado que pasará por el enlace también pueda atravesarlo este sin que se pierdan las condiciones sobre la QoS buscada. Finalmente aunque no sea una parte de asignación de recursos en la linea 16 se calcula el máximo delay posible con la siguiente ecuación:

$$T(u, v, q) = \frac{\sum A_B[u, v, q] 1_{p(u,v,l) \geq p(u,v,q)} + P_{max}}{S(u, v, q)} \quad (3)$$

Esta función quiere decir que en el peor de los casos para un paquete pasar de un nodo a otro por una cola q este tendra que esperar que sean enviados todos los paquetes en buffer que tengan una prioridad igual o mayor a la de él y en el caso de ser necesario que se complete alguna transmisión en curso la cual no pueda ser interrumptida.

IV. Resultados

Los siguientes son los resultados obtenidos en el paper [1], para las simulaciones se usa una topología típica de redes industriales como se ve en la figura 6

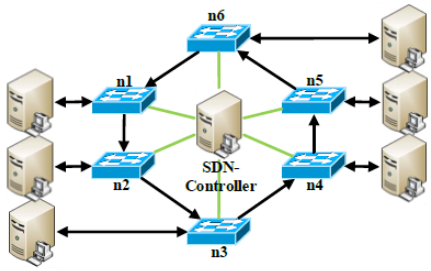

Fig. 6: Red de topología anillo, con 6 nodos de switcheo y 6 nodos de origen o destino

Se configuran los enlaces para que todos tengan 4 distintas colas de prioridad, cada enlace físico tiene una tasa de bits de 1[Gb/s] y cada cola tiene una capacidad de buffer de 90000[Bytes].

Cada paquete a enviar tiene un tamaño de 64[Bytes] y los resultados se obtuvieron a partir de enviar un tráfico variado de las distintos tipos de servicio, donde se usaron en total 968 combinaciones, el tiempo de ejecución de los algoritmos estudiados se pueden ver en la figura 8.

Service Class	Mean Bit Rate r_c [kByte/s]	Burst Size b_c [Byte]	Delay Limit t_c [ms]
$c = 1$	10	100	5
$c = 2$	10	100	10
$c = 3$	10	100	20
$c = 4$	10	100	50

Fig. 7: Parámetros usados

	Online Routing and Adm. Control	Offline Resource Allocation	MIP: Res. Alloc. and Routing
25th percentile	2.85 μs	24.9 s	148 s
mean	3.47 μs	27.1 s	333 s
75th percentile	3.95 μs	29.9 s	333 s

Fig. 8: Resultados obtenidos en comparación a método MIP.

Se puede ver que en promedio el tiempo de ejercución de la asignación de recurso, ruteo y admisión de control es $3.47[\mu s] + 27.1[s]$, esto se compara con el metodo MIP para ruteo y asignación de recursos el cual se demora en promedio 10 veces más.

Con respecto a otros parámetros:

- Utilización promedio de enlaces: En el modelo estudiado se obtuvo que se tuvó una utilización promedio de enlace del 93%, en cambió MIP se usa aproximadamente solo el 60% de la capacidad de los enlaces.
- Utilización promedio del buffer: En el modelo estudiado se obtuvo una utilizacion buffer promedio es menor al 92%, mientras que en el método MIP se usa practicamente el 100% de buffer, considerando que en el modelo estudiado se usa 93% del enlace y 92% de buffer se considera que es mejor que el MIP el que usa todo el buffer para una utilización menor del enlace.
- Se obtiene que el retardo promedio en el modelo estudiado es aproximadamente 40% del delay máximo permitido para este caso t_f, mientras que en MIP se obtiene que es un 60% de t_f.

V. Conclusión

A partir de los resultados que se obtuvieron en el paper considero que si bien estos presentan claramente una mejora al método MIP en este caso, no se da ningún tipo de garantía que en distintas topologias o quizas con más o menos nodos el método estudiado siga siendo más eficiente que el otro, sin embargo por lo estudidado sobre los algoritmos y el modelo propuesto se concluye que el método no debería tener problemas al ser implementado en cualquier tipo de red que use enlaces físicos.

Finalmente se concluye que los algoritmos estudiados cumplen con los requerimientos necesarios para generar una comunicación con QoS.

REFERENCES

[1] Jochen W. Guck, Martin Reisslein, and Wolfgan Kellerer, *Function Split Between Delay-Constrained Routint and Resource Allocation for Centrally Managed QoS in Industrial Networks*.

[2] L. Seno, F. Tramarin, and S. Vitturi, *"Performance of industrial communication systems: real application contexts," IEEE Industrial Electronics Magazine, vol. 6, no. 2, pp. 27–37, 2012.*

[3] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, *"Software-defined networking: A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.*

[4] D. S. Reeves and H. F. Salama, *A distributed algorithm for delay-constrained unicast routing,"*

[5] http://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/