Technical Report 2009-09-01
Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University
http://medianet.kent.edu/technicalreports.html

CRP Solver for Scheduling-based Forwarding in Intermittent Networks

Omar Y. Tahboub and Javed I. Khan
Media Communications and Networking Research Laboratory
Department of Math & Computer Science, Kent State University
233 MSB, Kent, OH 44242
otahboub| javed@kent.edu

I. Introduction

In this technical report, we first present a formal model of Intermittent Networks (INTNET). Constraint Resource Planning (CRP) algorithm has been applied to obtain efficient approximate solutions to NP-complete problems in operations research. CRP is a subclass of Constraint Satisfaction Problem CSP. The choice of CRP is due to its appeal of being generic execution shell. Moreover, CRP provides polynomial time bounded execution shell called CRP-engine, which employs both heuristics to achieve near optimum solutions. In this report we also present a CRP-based algorithm for data flow forwarding scheduling in INTNET. The performance studies to be included in different reports.

II. Concept: Intermittent Network (INET)

A. Topology Specifications

An intermittent network is represented by a ***Base Topology*** G and a *link intermittency function* (LIF). Base, topology $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{E})$, where $\boldsymbol{N} = \{n_1, n_2, \ldots, n_m\}$ be the set of m nodes and $\boldsymbol{E} = \{e_1, e_2, \ldots, e_n\}$ be the set of links, each link $e_i \in \boldsymbol{E}$ connects a pair of nodes $(n_u, n_v) \in \boldsymbol{N}$.

Each node $n_i \in \boldsymbol{N}$ is defined by the record $[c_i, b_i]$, where c_i denotes node's service rate (in bits per second) and b_i denotes it's storage capacity in bits. Each edge $e_i \in \boldsymbol{E}$, is defined by the record $(bw_i, l_i, s_i(t))$ where bw_i denotes the data link bandwidth (bps), l_i denotes the data link propagation delay (seconds) and the function $\boldsymbol{s_i(t)}$ *is the link intermittency function*.

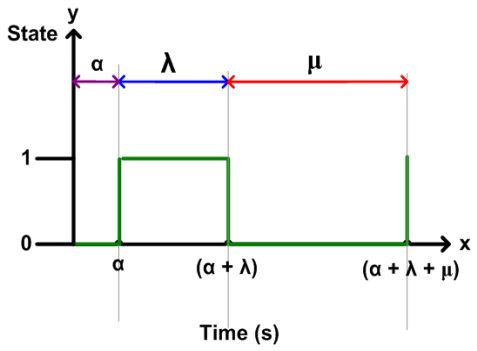

Fig. 1: The Link Intermittency Function LIF

In this paper, the link intermittency behavior is modeled as a periodic step function described by a triplet $[\boldsymbol{\alpha_i}, \boldsymbol{\lambda_i}, \boldsymbol{\mu_i}]$.Where, $\boldsymbol{\lambda_i}$ denotes the duration of *active contact* between the nodes (n_u, n_v), $\boldsymbol{\mu_i}$ denotes the duration of *post-contact inactivity*, and $\boldsymbol{\alpha_i}$ denotes the duration of the *pre-contact inactivity*. The period is given by $T_i = (\alpha_i + \lambda_i + \mu_i)$ Fig- 1 a LIF.

Due to intermittency, a link's effective bandwidth may not be the same as its data link capacity. The effective bandwidth B_i is defined as follows:

$$B_i = bw_i \times \frac{\lambda_i}{(\alpha_i + \lambda_i + \mu_i)} \quad (1)$$

Due to link intermittency, the real topology is time dependent and dynamic. The instantaneous topology $\boldsymbol{G}^{(t)}$ at any time is a sub-graph of $\boldsymbol{G}$, and comprised of all nodes and the link subsets which are in *active contact state* at the instant.

B. Data Flow Transfer Request Specifications

A network processes a set of transfer requests $\boldsymbol{T} = \{t_1, t_2, \ldots, t_n\}$. Each transfer requests t_i is defined by the record $(u_i, v_i, o_i, dl_i, s_i)$, where u_i is the source node, v_i is the destination node, o_i is the task origination time in seconds, dl_i is the desired task completion time or desired deadline in seconds, and s_i is the task size in bits. Note that o_i, dl_i and $s_i > 0$ and $dl_i > o_i$.

C. Route Specifications

A network route solver assigns a route r_i for each dataflow transfer requests $t_i \in \boldsymbol{T}$,. The route r_i is defined by a ordered set of k node hops $\{u_i, n_{i,2}, \ldots, n_{i,j}, \ldots, n_{i,(k-1)}, v_i\}$, or as *k-1* link (edge) hops $\{e_{i,1}, e_{i,2}, \ldots, e_{i,j}, \ldots, e_{i,k-1}\}$, where $e_{i,j}$ connects $n_{i,(j-1)}$ and $n_{i,j}$. A solution of a task t_i follows a route and aims to deliver within the delivery deadline.

The figure shown below utilizes an intermittent network $\boldsymbol{G}(\boldsymbol{N},\boldsymbol{E})$, where $\boldsymbol{N} = \{n_1, n_2, \ldots, n_{10}\}$ be the set of *m* vertices (nodes) and $\boldsymbol{E} = \{e_1, e_2, \ldots, e_{12}\}$ and further illustrates a route r_i committed to the task t_i. The task t_i represents a data flow transfer request between the nodes n_1 and n_{10}, while the route r_i is its corresponding solution given as $\{\{n_1, n_4, n_5, n_9, n_{10}\}, \{e_5, e_{11}, e_{19}, e_{24}\}\}$.

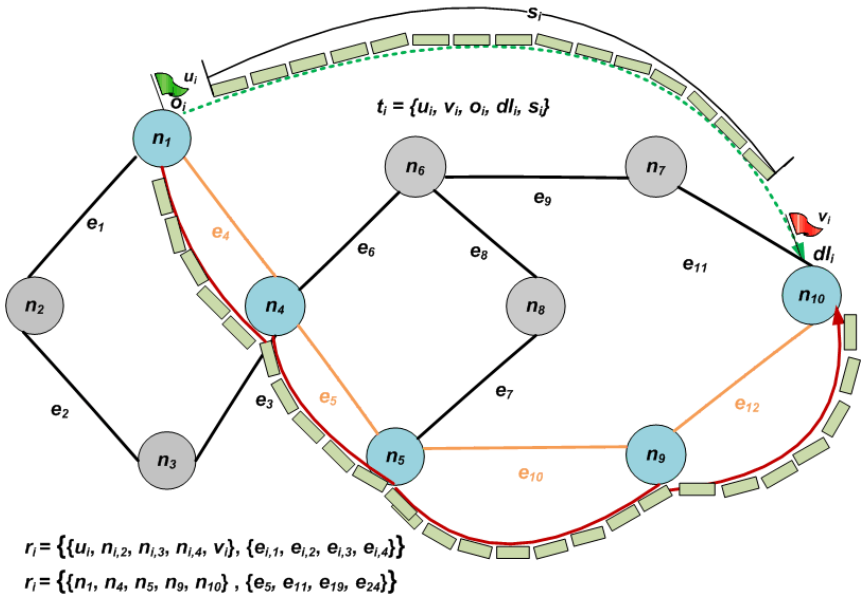

Fig. 2 Data Route Representation

D. Intermittency Pathway Specifications

Persistency & Simultaneity Criteria (PSC): In classical persistent networking (PNET) with multi-hop scenario, a pathway is recognized when all the links in a pathway are in active contact simultaneously. Let WC(λ_i) denotes the *wall clock time* function for link-hop e_i contact time. Under the wall of time function, the intersection relation is defined; two link-hops e_i and e_{i+1} in a route are said to be persistent concurrently if and only if WC(λ_i) ∩ WC(λ_{i+1}) ≠ Φ. A persistent route exists if and only if:

$$\bigcap_{e_i \in r} WC(\lambda_i) \neq \phi \tag{2}$$

The condition is also satisfied for fully persistent network or for the i^{th} link in the pathway has $\lambda_i = T_i$.

This PS criterion is however, overly restrictive and is not necessary for communication in general communication networks (such as transportation network etc.) including in intermittent network (INET).

Ordered Contact Criterion (OCC): In INET a path is recognizable even when only a subset of the links are simultaneously active are such as $\lambda_i < T_i$. Consider a pathway *r* consisting of *k* node-hops and *k-1* link–hops and LIF, a pathway would exist if the following two conditions hold. The wall clock time of contact state in any hop precedes that of all subsequent hops and wall of time of each two consecutive link-hops overlap. The first condition is shown by equation (3), while the other is shown by equation (4)

$$(WC(\lambda_i) \leq WC(\lambda_{i+1})) \leq WC(\lambda_{i+2}) .. \leq WC(\lambda_n) \tag{3}$$

$$\forall e_i \in r(WC(\lambda_i) \cap WC(\lambda_{i+1}) \neq \phi) \tag{4}$$

There is not trivial way to extend the routing algorithm developed assuming PSC to work under the generalized network with OCC. It is possible to show that a traffic routed with classical distributed shortest-path routing/forwarding might be stuck in oscillatory forwarding (*intermittent link maze)* may never reach its destination, although a perfect path may exist with OCC condition.

Communication under OCC criterion requires upgrade of both forwarding functionality and routing functionality of the network. Now in the next two sections we describe both.

E. Data Flow Decomposition and Scheduling

Forwarding function under OCC intrinsically requires the ability of data buffering at each hops. The possible extension to network architecture to accommodate forwarding for link intermittency may occur at three possible levels of network architecture. This is tied with the data unit abstraction levels in today's network. At the application level, the unit of data processing is the entire application data unit. At the network level, data flow is fragmented into the finite set of packets and a packet is viewed as the unit of data processing. While at the physical level (or streaming application level) bytes can be viewed as the unit of data processing. Correspondingly, the buffering operation for OCC can be implemented at any of these three levels but as we will show each will have distinct bounds for traffic efficiency and network resource requirements.

Given a data flow request *t* decomposed into as set of data packets {P_1, P_2,…, P_j,…, P_n }

streamed through an intermittent pathway as a *n-hop* route r = {*{u, n_2,…,n_k, n_{k+1},…, v} {e_1,e_2,…,e_k,…, e_{n-1}}*}. For the packet $P_j \in t$ forwarded by n_k to n_{k+1} via e_k, logical data unit schedule du_j is defined by the record (seq_j, sz_j, ps_j, pt_j, dep_j, arr_j, sld_j). The attributes seq_j is the sequence number; sz_j is the unit size in bits, ps_j processing starting time, pt_j *is* $_k$ is the du_j elapsed processing time at n_k, dep_j is the departure time from n_k, arr_j is the arrival time at n_{k+1}, and sld_j is the waiting time deviation incurred.

Moreover, for entire set of packets {P_1, P_2,…, P_j,…, P_n } and also forwarded form by n_k to n_{k+1} via e_k, the link-hop data flow transfer schedule is denoted by st_k = {du_1, du_2,…, du_j,…, du_n }. For the set of link-hops {e_1,e_2,…,e_k,…, e_{n-1}}$\in r$, the route forwarding schedule is denoted by *ut* ={st_1, st_2,…, st_j,…, st_{n-1} }.

$$b_k \leq (B_{k-1} + B_k) \qquad \blacksquare$$

III. Scheduling Algorithm for Data Flow Forwarding in (INET)

A. Constraint Resource Planning Route Scheduling Solution Framework

The Design of the proposed forwarding scheduling in (INTNET) is based on *Constraint Resource Planning CRP* approximation algorithms deign principle [7]. CRP is a subclass of Constraint Satisfaction Problem CSP. The choice of CRP is due to its appeal of being generic execution shell. Moreover, CRP provides polynomial time bounded execution shell called CRP-engine, which employs both heuristics to achieve near optimum solutions.

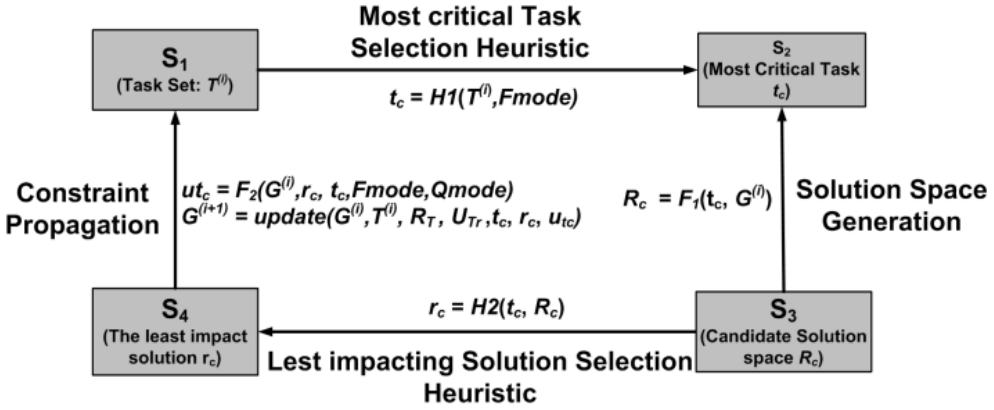

Fig.3: The CRP-based Solver Model

The CRP-Solver framework for the transit reservoir planning in (INTNET) problem is illustrated by the four-corner model shown below. The inputs to the solver includes the residual network $G^{(i)}$, the task set $T^{(i)}$, forwarding mode *Fmode* and the equilibrium mode *Qmode*. The outputs of solver are: route scheduled R_T, task schedule T_R, and forwarding schedule set U_{Tr}. Note that In addition, *Fmode* is an enumeration {*HOPPED*, *OVERLAPPED*, *EAGER*} and *Qmode* is also an enumeration {*FLAT*, *PROGRESSIVE*}.
It is shown that this solver model invokes four functionalities: task criticality (**H1**), solution cruciality (**H2**), solution space generation ($\mathbf{F_1}$) and constraint propagation ($\mathbf{F_2}$).

B. H1: Most Critical Task Evaluation

This heuristic selects of the most critical task according to the current state of $G^{(i)}$, the task set $T^{(i)}$ and forwarding mode constraint *Fmode*. The *criticality* of the task t_k in $T^{(i)}$ is the *completion earliness* when routed through its *Tentative Shortest Path (TSP)* r_{TSP}. Completion earliness is the difference between the task deadline the estimated *total transfer delay* via r_{TSP}, according to two cases. When (*Fmode* = *HOPPED*), the estimated completion earliness is $dl_k - \left(\sum_{e_i \in r_{TSP}} \left(\left(s_k / B_k \right) T_k \right) + o_k \right)$. When (*Fmode* = *OVERLAPPED* or *EAGER*), the estimated *total transfer delay* is size of t_c divided by the minimum block size in r_{TSP}, multiplied by the maximum link-hop periodic time step. Hence, the estimated completion earliness is $dl_k - \left(\left(s_c / \min(B_k) \right) \max(T_i) + o_k \right), \forall e_i \in r_{TSP}$. Finally the most critical task t_c one with minimum estimated completion earliness is($t_c = \min(\gamma_k), \forall t_k \in T^{(i)}$).

C. F_1: Solution Space Generation Function

Based on the most critical task t_c selected by H1, this function explores all possible routes (candidate solutions set R_c) between u_c and v_c (source and destination of t_c) satisfying the *Ordered Contact Constraint OCC* defined earlier

D. H2: Least Impact Solution Evaluation

Mainly, this heuristic selects the least impact solution (route) r_c from the computed candidate solution set R_c. The *cruciality* or candidate solution r_k in R_c is the least interfering route with candidate the Tentative Shortest Paths TSP of all tasks in $T^{(i)}$. Hence, the lest crucial route r_c is denoted by the route whose link-hops are minimally shared by other pending tasks. For this purpose, each link in e_k in $G^{(i)}$ is labeled according to the following. First for each distinct pair of nodes (u, v), compute it's TSP $r_{(u,v)}$. Second, for each link-hop e_j in $r_{(u,v)}$ matching a task t_k in $T^{(i)}$ increment the cruciality (γ_k) by one. Finally, select the route r_c whose total link-hop cruciality is minimal.

E. F_2: Constraint Propagation Function

In order to commit r_c to t_c, the forward schedule ut_c defined in Section 2 is computed. The function F_2 acquires $G^{(i)}$, t_c, r_c, *Fmode* and *Qmode*, and generates ut_c. Note that *Fmode* and *Qmode* describe the forwarding mode and the data flow equilibrium constraints. Due to its relevance, we elaborate F_2 through pseudo code shown by Figure 3. The main loop builds the forwarding schedule $\boldsymbol{ut_c}$ of $\boldsymbol{t_c}$ through the $n - 1$link hops in $\boldsymbol{r_c}$ according to two cases. First, when ($k = 1$), the data flow transfer schedule $\boldsymbol{st_1}$ for $\boldsymbol{e_1}$ is computed using the heuristic the source link-hop scheduling heuristic $\mathbf{H_1}$. The inputs to this heuristic are inputs are t_c, r_c, n_k, (forwarding node-hop) n_{k+1}, (recipient node-hop), e_k (forwarding link-hop), link-hop e_{k+1}(next forwarding) and *Qmode*, while the output is the transfer schedule st_k. Second, when$1 < k < n$, the data flow transfer schedules $\{\boldsymbol{st_2, st_3,\ldots, st_k,,\ldots, st_{n-1}}\}$ for the set of link-hops $\{\boldsymbol{e_2, e_3,\ldots, e_{n-1}}\}$ are computed by the heuristic $\mathbf{H_2}$. This heuristic is called intermediate link-hop scheduling, whose inputs are also t_c, r_c, n_k n_{k+1}, e_k, e_{k+1}, *Qmode* and *Fmode*, while the output is the transfer schedule st_k. Finally, on the basis of the CRP solver framework, we devised three CRP-based solvers are defined in Table 1.

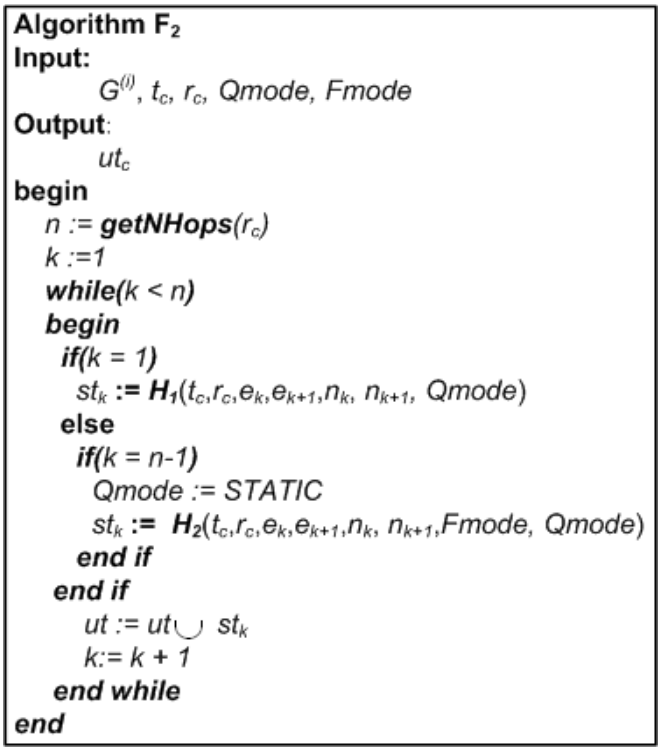

Fig. 4: The Constraint Propagation Function

The complete technical details of the CRP-solver and forwarding scheduling heuristics H1 and H2 are given the proceeding discussion.

a. Heuristics H_1: Source Link-Hop Forward Scheduling

This heuristic, computes st_1 the data flow transfer schedule initiated by n_1 to n_2 via e_1. The inputs are task ***t,*** route ***r,*** the source node-hop n_1, second node-hop n_2, source link-hop l_1, second link-hop l_2, and equilibrium mode ***Qmode***, while the output is the forward schedule ***st***. The variables Ω, Ψ, denote the local clock time at n_1 and relative time offset of the link hop l_1 LIF. Static and dynamic data flow equilibrium are performed by the functions ***doStaticFlowEq*** and ***doDynFlowEq.*** In addition node buffer reservation and release are performed by the functions ***rsvNodeBuff*** and ***rlsNodeBuffer***. Further, link capacity reservation is performed by the function ***rsvLinkCap***.
Note that the initialization step sets *i*, $k = 1$, Ω = o (task *t* origination time), ***m*** = s (size of the task), and $st = \Phi$. Next, the entire task is buffered at n_1 using the function starting at time Ω. Finally, The main loop computes the $st = \{du_1, du_2, \ldots, du_n\}$ through the following steps. The detailed steps of the heuristics are presented by Figure 2(b).

b. Heuristics H_2: Intermediate Link-Hop Forward Scheduling

The inputs are task ***t,*** route ***r,*** forwarding node-hop n_s, recipient node-hop n_d, forwarding link-hop ***l,*** next forwarding link-hop l_{next}, forwarding mode ***Fmode*** and equilibrium mode ***Qmode***, while the output is the transfer schedule ***st***. The variables Ω, Ψ, Δ denote the local clock time at n_1, relative time offset of the link hop ***l*** LIF and the data packet waiting slide delay due to The initialization step sets $k = 1$, and $st = \Phi$, $N = |st'|$ (the number of data packets relative to task size ***s*** and the block size ***B***), and $\Omega = arr'_n$, (the arrival time of P'_n given by du'_n), Based on ***Fmode*** the value of is Ω determined if *Fmode* is hopped $\Omega = arr'_1$, (the arrival time of P'_1

given by $\boldsymbol{du'_1}$), otherwise $\boldsymbol{\Omega}$ is determined later. The main loop computes the $\boldsymbol{st} = \{\boldsymbol{du_1}, \boldsymbol{du_2}, \ldots, \boldsymbol{du_n}\}$ in N iterations. In the i^{th} iteration ($1 \leq i \leq N$) , the forwarding schedule for data unit du'_i buffered at n_s is computed.

F. CRP-based Forwarding Scheduling in INTNETs

Based on the CRP solver framework, we devise three solvers namely, CRP-Hopped, -Overlapped, and Eager oriented towards solving the transit reservoir problem in (INTNET) using the hopped, overlapped and eager modes respectively. Based on the CRP solver model described earlier, each of the three solvers are defined through **H1**and **H2** heuristics, $\mathbf{F_1}$and $\mathbf{F_2}$ functions, and the data flow equilibrium mode. Therefore, each of the devised CRP-based models has two versions: one with flat time equilibrium and another with time progressive equilibrium.

Solver	Most Critical Task Selection	Solution Space Generation	Least Impacting Solution Selection	Constraint Propagation
CRP-Hopped	**H1**(HOPPED)	$\mathbf{F_1}$	**H2**	$\mathbf{F_2}$(HOPPED,STATIC)
				$\mathbf{F_2}$(HOPPED, DYNAMIC)
CRP-Overlapped	**H1**(OVERLAPPED)	$\mathbf{F_1}$	**H2**	$\mathbf{F_2}$(OVERLAPPED, STATIC)
				$\mathbf{F_2}$(OVERLAPPED, DYNAMIC)
CRP-Eager	**H1**(EAGER)	$\mathbf{F_1}$	**H2**	$\mathbf{F_2}$(EAGER, STATIC)
				$\mathbf{F_2}$(EAGER, DYNAMIC)

The CRP-based Solvers

The total data size to be forwarded from n_s to n_d D is initialized to sz'_i. Next, the time counter is $\boldsymbol{\Omega}$ adjusted to $arr'_{k,}$. The inner loop instantiates the set of data units corresponding to du'_i whose size is D bits.
Finally, the complete details of the utility function G1, G2 and G3 are shown by Fig.6 below.

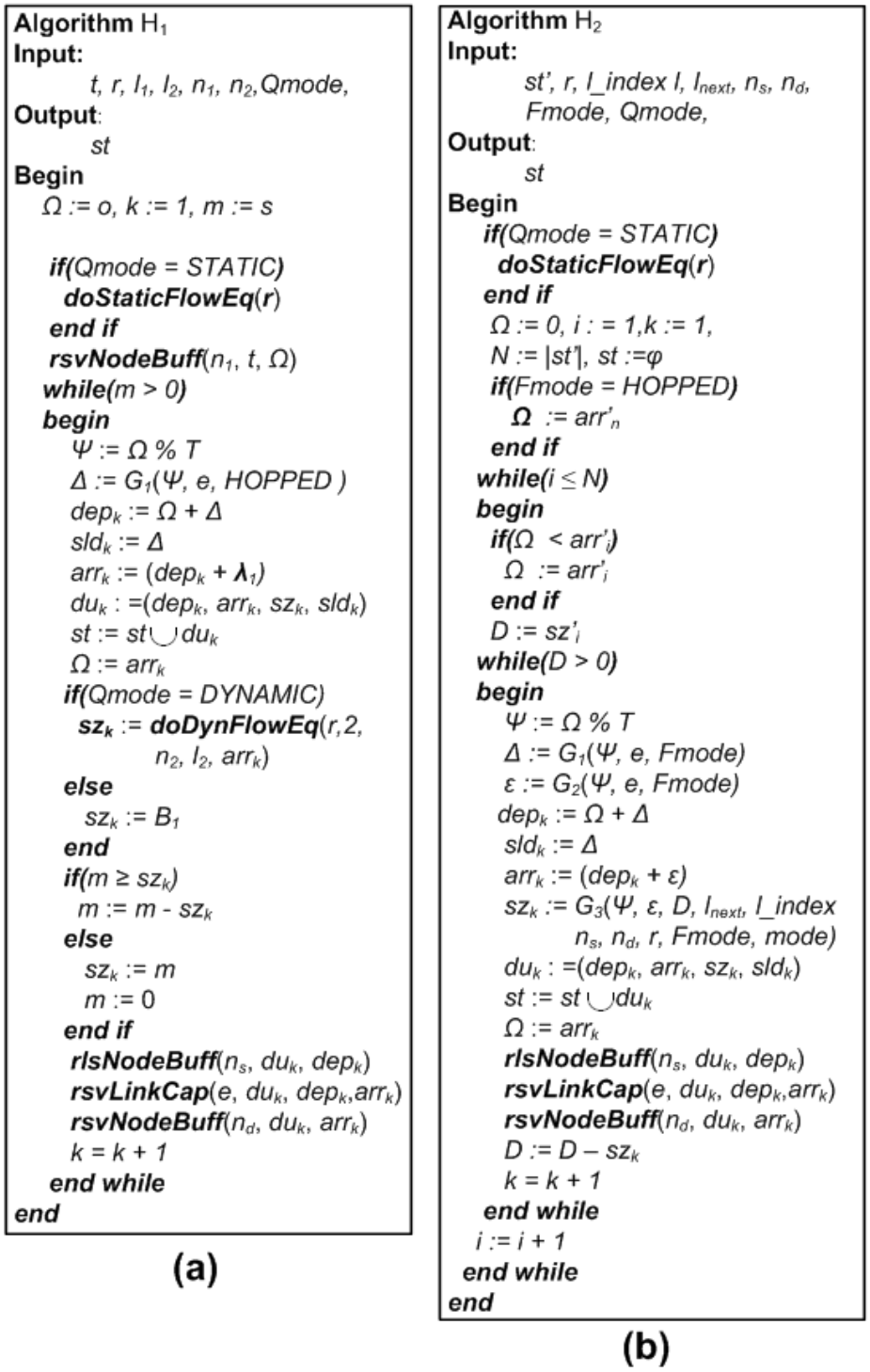

Fig.5: The H_1 and H_2 Forwarding algorithms

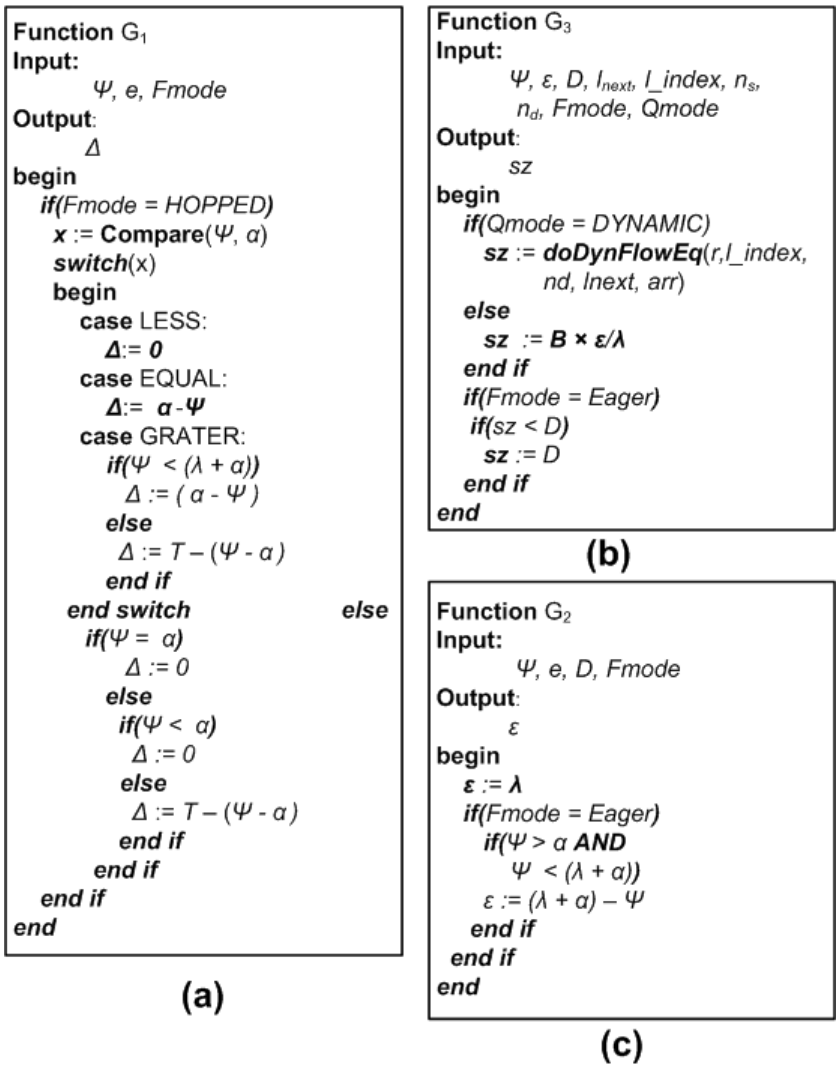

Fig.6: The utility functions (a) G_1, (b) G_2 and (c) G_3

IV. Performance Evaluation

The complete performance of three proposed CRP-based solvers is presented in [11].

V. References

[1] S. Jain, K. Fall, and R. Patra,, "Routing in a Delay Tolerant Network", Proc of ACM SIGCOMM, 2004, pp. 145 – 158.

[2] A. Di, Nicolo, and P. Giaccone, "Performance limits of real delay tolerant networks", WONS 2008. pp. 149- 155.

[3] E. P Jones, L. Li, and P. A.Ward, "Practical routing in delay-tolerant networks",
In Proc 2005 ACM SIGCOMM Networking, , Pennsylvania, USA, August 26 - 26, 2005,

[4] J. Khan, and O. Tahboub, "A Reference Framework for Emergent Space Communication Architectures oriented on Galactic Geography", Proc AIAA Space Operations '08, Heidelberg, 2008.

[5] A. Balasubramanian, et al.,"Interactive WiFi Connectivity or Moving Vehicles", Proc. ACM SIGCOMM, vol. 38, Issue 4, 2008, pp. 427-438.

[6] P. Johansson, M. Kazantzidis, R. Kapoor, M. Gerla, , "Bluetooth: an enabler for personal area networking", IEEE Network Magazine, vol. 15, issue 5, 2001, pp. 28 – 37.

[7] N. P. Keng and D Y. Y. Yun, " A planning/scheduling methodology for the constrained resource problem", Proc. of IJCAI 89, 1989, pp. 998-1003.

[8] O. Tahboub, and J. Khan, "Routing Scheduling in Intermittent Networks", Tech. Report, Medianet Lab, Computer Science Department, Kent, Ohio, 2009.

[9] K. Fall, "A delay-tolerant network architecture for challenged internets", Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, 2003, pp. 27 – 34.

[10] J. Khan, O. Tahboub, "DCN@MPLS: A Network Architectural Model for Dynamic Circuit Networking at Multiple Protocol Label Switching," in Proc SAINT, 2009 Seattle, WA, pp.267-270.

[11] Tahboub, O., Khan, J., **"Transit Reservoir Planning in Intermittent Networks",** submitted to IEEE ITNG 2010, Las Vegas, Nevada.