network

MDPI

Article

A Cognitive Anycast Routing Method for Delay-Tolerant Networks

Ricardo Lent

Department of Engineering Technology, University of Houston, Houston, TX 77204, USA; rlent@uh.edu

Abstract: A cognitive networking approach to the anycast routing problem for delay-tolerant networking (DTN) is proposed. The method is suitable for the space–ground and other domains where communications are recurrently challenged by diverse link impairments, including long propagation delays, communication asymmetry, and lengthy disruptions. The proposed method delivers data bundles achieving low delays by avoiding, whenever possible, link congestion and long wait times for contacts to become active, and without the need of duplicating data bundles. Network gateways use a spiking neural network (SNN) to decide the optimal outbound link for each bundle. The SNN is regularly updated to reflect the expected cost of the routing decisions, which helps to fine-tune future decisions. The method is decentralized and selects both the anycast group member to be used as the sink and the path to reach that node. A series of experiments were carried out on a network testbed to evaluate the method. The results demonstrate its performance advantage over unicast routing, as anycast routing is not yet supported by the current DTN standard (Contact Graph Routing). The proposed approach yields improved performance for space applications that require as-fast-as-possible data returns.

Keywords: cognitive networking; delay-tolerant networks; spiking neural networks; neuromorphic computing

Citation: Lent, R. A Cognitive Anycast Routing Method for Delay-Tolerant Networks. *Network* **2021**, *1*, 116–131. https://doi.org/10.3390/network1020008

Academic Editor: Amitava Datta

Received: 15 June 2021
Accepted: 28 July 2021
Published: 30 July 2021

1. Introduction

With anycast routing, data can be delivered to (or from) any one of equally suitable sinks (or sources) by establishing a one to any-of-many relation. When correctly used, it can help to simplify network services while improving the data delivery performance. Practical uses of anycast routing today can be found mainly for the terrestrial domain. A common application involves information retrieval in content-delivery networks (CDN), where user requests can be served from the most appropriated data center hosting a geographically replicated service [1–4]. This alternative to unicast routing helps to reduce service latency that can originate from normal or intended network congestion, e.g., as produced by a denial-of-service attack [5]. In addition, it helps improve service resiliency to partial network outages [6]. Looking beyond 5G, anycast routing can serve as an efficient data delivery mechanism for integrated local, cellular, and satellite networks to handle massive and high-frequency data collection and dissemination tasks.

Despite not being previously explored in detail, anycast routing may help to mitigate some of the challenges involved in space networks [7,8]. For instance, consider a constellation of LEO satellites [9] providing intermittent communication services for a large number of sensor nodes of a smart environment (e.g., smart ocean [10]). The data from the nodes could be forwarded to different sinks through diverse paths, which may be automatically selected by the anycast service based on the location, network congestion levels, and satellite link availability. Similarly, consider a high throughput satellite (HTS) system with site diversity [11,12], where the ground gateway sites are interconnected via high-speed links. The use of the millimeter wave spectrum (mmWave) and free-space optical (FSO) in the feeder links makes the communication channels highly susceptible to atmospheric phenomena [13]. In this case, anycast routing could help to automate the traffic distribution among multiple feeder links and mitigate the impact of adverse weather

affecting one or more gateway sites. As a third example, consider a space–ground sensor network with limited contact opportunities. In this scenario, it is permissible to expect that the data collected from the space sensors could be delivered to any ground station for processing or their central collection using ground channels. These are a few examples that show the potential benefits of the integration of anycast routing and delay-tolerant networking (DTN) [14].

This work explores how to achieve (near) optimal anycast routing on a space–ground DTN. The optimality of the routing is defined with respect to a metric of interest, commonly the response time of the data bundles. The main features of the network involve regular but mostly predictable contact opportunities, time-varying channel conditions that may not be observable network-wide by all nodes, and asymmetric and large propagation delays. The Consultative Committee for Space Data Systems (CCSDS) standard on space DTNs is Schedule-Aware Bundle Routing (CCSDS, 734.3-B-1), which uses Contact Graph Routing (CGR) to determine unicast paths [15]. However, the standard does not define anycast routing capabilities. Additionally, a subset of the network nodes may be geographically dispersed and subject to connectivity losses for extended periods, and therefore network partitions are possible. This feature conveys the general lack of knowledge about network-wide conditions, such as the actual buffer occupancies and the channel bit error rates, which creates additional challenges to the identification of the optimal paths from a bundle delivery time perspective.

The proposed method builds on top of the Cognitive Space Gateway (CSG) [16] approach to unicast routing that has been proposed for space DTNs. A spiking neural network (SNN) and learning method are used to formulate the Cognitive Network Controller (CNC) [17,18], which is used as the learning element of an autonomic loop. The CNC decides the optimal anycast routing option for data bundles and the expected utility (or cost) of such routing decision helps to rebalance the weights of the SNN and to induce a possible switch to a different routing alternative in the near future. The main advantages of the proposed anycast method is that routing is both decentralized and dynamic. The CNC design is also suitable for neuromorphic hardware, and therefore involves ultra-low power requirements, e.g., when using Intel's Loihi chip [19]. Because the autonomic loop continuously maps the routing decisions and the expected costs, the CSG is able to quickly adapt the next routing decisions to better exploit the parallel resources that may be available to reach any member of the designated anycast group.

The remainder of the paper is organized as follows. The next section discusses related works. Section 3 offers details of the CSG and the proposed anycast routing method. Details of the experimental evaluation environment are given in Section 4 and the performance measurements of the proposed anycast routing method are reported in Section 5. Finally, Section 6 provides concluding remarks.

2. Related Works

Prior works have investigated anycast for regular networks, i.e., holding the original Internet-design assumptions. Anycasting is supported by both versions 4 and 6 of the Internet Protocol (IP) as documented in RFCs 1546 and 4291, and the Border Gateway Protocol (BGP-4, RFC 4271). It plays a vital role for DNS and CDN and services, but different issues have been noted, including the detection latency of IP anycast prefixes [20], AS-level anycast path inflation [21], and client-server mapping limitations that arise in CDN routing [2]. Several enhancements have been discussed focusing on either performance, see, e.g., [22,23], or security, e.g., to mitigate distributed denial-of-service (DDoS) attacks [5]. It has been shown that anycast routing can achieve near-optimal response times and that it can be tailored to suit the QoS requirements of DiffServ networks [24], but these techniques often require knowledge about the global network state (i.e., router loads) [25,26]. Anycast services can also be used at other layers of the protocol stack. Representative examples include methods to achieve low-latency access for edge computing services [27], to overcome the high packet loss rates of Internet-of-Things (IoT) environments [6], to optimize

the selection of replicated web servers [28], to improve the reservation system for electric vehicles and reduce the wait time [29], and to reduce the energy consumption of data centers [30] and wireless sensor networks [31,32].

However, DTNs and space networks are characterized by the lack of Internet-design assumptions, and therefore the works discussed above may not be directly applicable. One of the issues is the lack of a permanent end-to-end connectivity among all nodes [33]. Currently, the literature on anycast DTN routing is sparse. A group of works has addressed the probabilistic DTN case with random contacts. Along that line of research, the anycast social distance metric (ASDM) was introduced [34]. The ASDM helps to direct routing messages in the direction of the location where most group members are expected to be present. In the time constrained anycast (TCA) method [35], the one-hop and two-hop delivery probability to the anycast group are determined from the distribution of inter-contact times (ICT). Routing decisions are then made based on an exponential distribution of ICTs. An extension to CGR has also been proposed [36], which considers the history of the observed contacts.

Theoretical studies of the problem include the application of epidemic routing to DTN anycasting, which has been studied under a Markovian assumption [37]. The semantics related to the address identifier to be used in DTNs have been examined at least for the case without recurrent network partitioning [38]. Larger anycast groups increase the chances of delivering the information quickly to any of the group members. This case was studied in simulation for a sparse mobile network communicating ad hoc and using the receiver-based forwarding (RBF) approach [39]. A genetic algorithm (GA) has been applied to DTN anycast routing for deterministic networks [40], showing reduced average delays compared to the shortest path but mainly with a small number of concurrent sessions, possibly because of the slow convergence time of GAs.

3. Anycast Routing

The network is identified by a topology that is dynamically changing due to node mobility, scheduling decisions, and signal propagation issues. The changes mainly follow well-defined patterns, e.g., resulting from orbital mechanics, and therefore, the future contact times between nodes can be predicted. The service unit is called *bundle* and the use of multiple contacts and nodes may be needed to deliver each bundle to the sink. Any node can operate as a gateway routing bundles for other nodes. It is assumed that the one-hop transmissions are provided by a reliable convergence layer protocol and the use of the Licklider Transmission Protocol (LTP) is assumed. The routing process aims to deliver each bundle both efficiently and with a minimum delay to any member node of the anycast group. The former aim prevents the use of bundle duplication, and therefore any form of epidemic routing, whereas the latter requires the optimization of both the member selection and the path to reach that node. Furthermore, the routing decisions must be done autonomously without any central coordination. The gateways are not aware of the global network state. The main concepts behind the cognitive space gateway approach, which was previously introduced for unicast routing, are concisely discussed followed by the proposed anycast extension.

3.1. CSG Routing Method

Routing decisions are determined independently by each gateway with the help of an SNN architecture (i.e., the cognitive network controller—CNC [18]). After choosing the optimal outbound link for each data bundle, the SNN is modified so that the CNC can adapt to the expected outcome of such a decision (i.e., the routing cost) and mitigate the performance impact of the latest bundle transmission to the next bundles. This process is achieved by modifying some of the synapse strengths of the SNN. To illustrate the process, a brief overview of the SNN model is first provided.

3.1.1. Spiking Neuron Model

The CNC uses the leaky-integrate-and-fire (LIF) spiking neuron model, although other spiking neuron models are also suitable [41]. The LIF model describes the electrical properties of biological neurons as given by a resistor–capacitor (RC) circuit that is driven by a current $I(t)$. The membrane potential is a function of time $u(t)$ that can be determined with the expression: $\tau \frac{d}{dt} u(t) = -u(t) + RI(t)$, where $\tau = RC$ is the selected time constant of the circuit and $I(t)$ the total stimulus.

As in the biological case, neurons can communicate with each other via nerve impulses or *spikes*. One spike is emitted by a neuron when its membrane's potential reaches a given threshold θ. A neuron produces spikes at times $t^{(f)} : u(t^{(f)}) = \theta$. After emitting one spike, the membrane potential drops significantly, i.e., the neuron requires to rest for some time. The potential drops to the level $u_r < \theta$ for a given time period Δ (i.e., the refractory period). During that time, the neuron ignores the effect of new spike arrivals or any external stimulus. The SNN connections determine which neurons will be affected by the emission of spikes and by how much as given by the values of the synaptic strengths (or connection weights). From neuron j (the *pre synapse*) to neuron k (the *post synapse*) the synapse strength is denoted by w_{jk}, with $w_{jk} \leq 0$. Spiking neurons are either of excitatory or inhibitory nature. In the former case, the spikes emitted by such neurons tend to increase the membrane's potential of the post synapse neuron. In the latter case, the effect is the opposite.

The stimulus $I(t)$ consists of the aggregated effect of the potential gradients produced by all of the neuron's dendrites, which include the effect of any external stimulus $i_e(t)$ and the spikes arriving from neuron j. The spikes arriving at the post-synaptic neuron are modeled by the impulse train $i_{jk}^{(f)}(t) = \delta(t - t_j^{(f)} - d_{jk})$, which occur at the firing times f, delayed by d_{jk}. Because more than one synapse may connect any two neurons, the notation emphasizes that the effect of the k-th synapse originated at neuron j. Therefore, the resulting stimulus can be expressed as $I(t) = i_e(t) + \sum_f \sum_j \sum_k w_{jk}\, i_{jk}^{(f)}(t)$.

3.1.2. Cognitive Network Controller

The CNC [18] defines an SNN architecture of $2(K-1)$ neurons, where K is the number of outbound link decisions (also known as actions) that are available for the bundle. Each action is represented by one excitatory neuron: $N_1, \ldots, N_K$ connected as a mesh with synapses w_{N_i,N_j} in both directions for any two neurons $i, j = 1, \ldots K$, $i \neq j$. The goal of the other neurons is to provide membrane potential regulation as the excitatory signals of the core neurons can lead to potential saturation. This is achieved by making the neurons $M_1, \ldots, M_{K-2}$ inhibitory with directed synapses toward the core neurons: w_{M_i,N_j}, $i = 1, \ldots, K-2$, $j = 1, \ldots, K$.

To make a routing decision, the SNN is started by applying identical external stimulus I_{ref} to all neurons. The SNN activity is then observed to determine the time of emission of the neurons' second spike. The first spike is used for bootstrapping the SNN as that spike carries the effect of the current synaptic connection values to the post-synapse neurons. Once the neuron that emits the earliest second spike is detected, the routing decision is given by the link being represented by that neuron, i.e., the CNC decides to use the link $a = i^*$, provided that $N_{i^*} = \text{argmin}_{N_i}\ t^{(2)}(N_i) \quad ; i = 1, \ldots, n$, where $t^{(f)}(x)$ is the firing time of the f-th spike of neuron x. If multiple neurons are detected to emit their second spike at the same time, the link decision is carried out randomly among those options. Furthermore, to prevent local minima, the routing decisions are modulated by a random walk with a (small) probability of P_w.

3.1.3. CNC Learning

After a routing decision a has been made, the CNC computes the estimated cost (or negative reward) of that choice. In this paper, the interest is in optimizing response times, and thus the routing cost C is computed in terms of the expected delivery time of the

bundle (at the destination) via the selected outbound link. The average observed cost can be iteratively calculated using the expression: $G \leftarrow \alpha C + (1-\alpha)G$, where $0 \leq \alpha \leq 1$ is a hyperparameter. The gradient $\delta = G - C$ is used to update the SNN weights using the scaling factor $\eta > 0$ (the learning rate):

$$\begin{array}{ll} w(N_j, N_a) & \leftarrow w(N_j, N_a) + \eta\ \delta\ ; j = 1, \ldots, K;\ a \neq j \\ w(M_k, N_a) & \leftarrow w^k(M_a, N_a) - \eta\ \delta\ ; k = 1, \ldots, K-2 \end{array} \quad (1)$$

As routing decisions are made and the SNN's weights are changed, the range of weight values is observed. When the range exceeds a predetermined threshold, a weight regulation process is applied to shift and scale the weights to a selected range. This is done to prevent synapse connections from becoming too large or too small, which may impact the emission of spikes. The learning algorithm has the linear complexity $O(n)$, where n is the number of decisions.

3.2. Anycast CSG Routing

The proposed anycast routing method defines both the semantics of the anycast group and the mechanisms needed to optimally forward the bundles to any one member of the group. More than one anycast group may concurrently work. Consider one such group $A = \{k\}$, where k is a CSG node address.

3.2.1. Anycast group semantics

To fulfill the former requirement, a CSG gateway needs to determine whether it is part of A, which can be accomplished by defining the destination address d either as a compounded identifier or as a reserved identifier for the anycast group. The task is facilitated by the use of string identifiers by the CSG to name network nodes. In the first case, $d = A$, which offers the advantage that the source can dynamically modify the group composition without needing to communicate the details to other nodes, but requires extending the header size of the bundles with an overhead that is proportional to $|A|$. In the second case, $d = a$, where a is a string that identifies the group A. The size of a can be much shorter than $|A|$ but the method requires informing A to each $k \in A$, and so any group membership modifications are more involved than with the first method. In either case, the anycast group members need not confirm their membership to the source node. For greater flexibility, the CSG anycast method supports both methods so that the decision of which method will be used would depend on the features and the logistic constraints of specific missions.

3.2.2. Optimal Anycast Bundle Forwarding

Each unicast or anycast destination is assigned a separated CNC at any CSG gateway i. The reward-shaping method used by the CSG [16] to estimate the cost C for each routing decision toward destination d (i.e., the negative reward) requires three terms:

$$C = T_{i,j} + T_{j,d} + D_{i,d} \quad (2)$$

where $T_{i,j}$, is the average bundle *communication time* over the i, j link, $T_{j,d}$ is the communication time from gateway j to destination d, and $D_{i,d}$ is the expected *dormant time*, i.e., the total disruption time of all links along the path. The first two terms include the radiation and buffering times but not the contribution of link disruptions. It is convenient to define $T_{i,d} = T_{i,j} + T_{j,d}$. The bundle transmission time to the neighbor j can be measured after the event either by the convergence layer or through the single-hop bundle acknowledgement that is sent by the neighbor. In either case, gateway i keeps the average observation using: $T_{i,j} \leftarrow (1-\alpha)T_{i,j} + \alpha t_{i,j}$, where $t_{i,j}$ is the new measurement.

The single-hop bundle acknowledgements are also used by neighbor j to share its own estimation $T_{j,d}$ with the preceding node. The process repeats along the path with $T_{d,d} = 0$. Gateway i stores the value $T_{j,d}$ for the next calculation.

As with unicast destinations, the gateway determines the dormant time introduced by link disruptions by analyzing the contact plan. The contact plan is given as a list of tuples (t_s, t_e, n_s, n_e, f), where t_s and t_e are the start and end times of the contact between nodes n_s and n_e with expected channel features f. The latter term is ignored in this discussion for simplicity. The contact plan is updated by replacing $n_e \leftarrow a$. The feasible paths are then calculated with respect to a (i.e., considering a as a *supernode*) using Algorithm 3 and after defining the graph from the contact plan with the procedures indicated by Algorithms 1 and 2. Algorithms 2 and 3 are identical to the one used for unicast addresses except for the last step of Algorithm 3. The time complexity of the algorithms is quadratic in the size of the contact plan.

Algorithm 1 Prepare contact plan.

```
1:  procedure PREPAREPLAN(P', t, n_s, a, A)   ▷ P': contact plan, t: wall time, n_s: source, a:
    anycast identifier, A: anycast group
2:      P ← [ empty list ]
3:      P ← P + (n_s, n_s, t, ∞)                          ▷ append source contact
4:      P ← P + (a, a, t, ∞)                              ▷ and destination contact
5:      for all (n_i, n_j, t_s, t_e) ∈ P' do
6:          if t_e < t then
7:              continue                                  ▷ skip this entry
8:          end if
9:          if t_s < t then
10:             t_s ← t
11:         end if
12:         if n_j ∈ A then
13:             n_j ← a
14:         end if
15:         P ← P + (n_i, n_j, t_s, t_e)                  ▷ append entry
16:     end for
17:     return P
18: end procedure
```

The last step of Algorithm 3 filters the routes so that the list to be returned only includes as the next hop the optimal neighbor j^* (as selected by the SNN) and the learning process is done with reference to that node. Cost C is given by the lowest value τ that is returned by Algorithm 3. The cost value is then used to adjust the weights of the SNN as defined by the learning method. Additionally, a simple extension needs to be added to the normal processing of incoming bundles. When a gateway k receives a new bundle with destination address d, it needs to check for anycast group membership when d is an anycast identifier ($a = k$) (or $k \in A$ with the other semantic). If so, it passes the bundle to the upper layer.

Algorithm 2 Create contact graph

procedure CONTACTGRAPH(P) ▷ P: contact plan
  $G \leftarrow$ { empty associative array }
  $n \leftarrow$ length of P
  for $i \leftarrow 1, n$ **do**
    $(n_i, n_j, t_s, t_e) \leftarrow P[i]$
    $G[i] \leftarrow$ [empty list]
    for $j \leftarrow 1, n, i \neq j$ **do**
      $(n'_i, n'_j, t'_s, t'_e) \leftarrow P[j]$
      if $n_j == n'_i$ **then**
        if $t_s < t'_e$ **then**
          $G[i] \leftarrow G[i] + j$
        end if
      end if
    end for
  end for
  return G
end procedure

Algorithm 3 Search feasible routes

procedure GETROUTES(G, P, n_s, n_d, t, j^*) ▷ G: contact graph, P: contact plan, n_s: source, n_d: destination, t: wall time, j^*: reference neighbor
  $R \leftarrow$ [empty list]
  $W \leftarrow$ [empty list]
  $W \leftarrow W + ([0], [n_s], t)$ ▷ [0]: contact ID path, ▷ $[n_s]$: node path, 0: path cost, t: wall time
  while W is not empty **do**
    $C, N, \tau \leftarrow pop(W)$ ▷ get & del first entry
    $h \leftarrow$ last contact ID listed in C
    $(n_i, n_j, t_s, t_e) \leftarrow P[h]$
    for all $k \in G[h]$ **do**
      $(n'_i, n'_j, t'_s, t'_e) \leftarrow P[k]$
      if $k \in C$ or $n_j \in N$ **then**
        continue ▷ avoid loop
      end if
      if $k == 1$ **then** ▷ destination contact
        $R \leftarrow (C + k, N, \tau)$
      else
        if $\tau > t'_e$ **then**
          continue ▷ contact ended
        else if eta $< t'_s$ **then**
          $\tau \leftarrow t'_s$ ▷ wait for contact
        else if $\tau \leq t'_e$ **then**
          $W \leftarrow W + (C + k, N + n'_j, \tau)$
        end if
      end if
    end for
  end while
  return R sorted by cost (τ) and filtered so that for all $r \in R$ the next hop is j^*.
end procedure

4. Evaluation Testbed

The proposed anycast routing method was evaluated experimentally for a representative set of scenarios of space, air, ground, and sea. To this end, a network testbed of 10 Linux servers was deployed (PowerEdge Dell R220, Intel Xeon CPU E3-1270 v3 running at 3.50 GHz with 16 GB RAM). A machine virtualization hypervisor (VirtualBox 6.1) was used to create the virtual machines (VM) containing the anycast gateway software. The VMs were configured with 2 CPUs and 12 MB RAM without any CPU execution cap. Moreover, to prevent any performance degradation from multitenancy or the shared use of machine resources, a single gateway was created per host with direct access (i.e., using the *bridged adapter*) to each of the network ports. Each server was equipped with six network interface cards (NIC) in total, which were provided by a PCI express four-port NIC in addition to the two NICs embedded on the mainboard. The network topology is depicted in Figure 1. The edges indicate the one-way propagation delays used for scenario 2, but the topology was the same for all of the tests. The testbed was build using point-to-point connections with a nominal transmission rate of 1 Gbps (i.e., UTP cables connecting NICs). The unicast routing performance of the implementation has been experimentally compared to that of ION-DTN [42,43], which suggests the correctness of the system implementation.

A tool was developed to achieve dynamic and repeatable control of the topology links' state during each experiment run, which allows testing different network conditions. The Network Environment Emulator (NEE) allows programmatic control of the link delays, rates, and packet drops through a distributed, master–slave architecture that works on top of Linux's traffic control, where the master node calculates the intended state of the network over time and communicates the changes when they are due to the relevant clients (agents) that run on the gateway nodes. The main task of the agents is to implement the desired changes. For example, the deactivation of the link (i.e., link disruption) is achieved by applying a filter (*netem qdisc*) to the desired interface that drops 100% of the packets received. A contact is emulated by removing such filters. Likewise, changes in the bit error rate (BER) and transmission rates of the links can be programmatically controlled by the NEE. In addition to implementing link impairments, the NEE also works as an information source for the contact plan, which is calculated for a given future window (60 s in the tests) and communicated to the gateways.

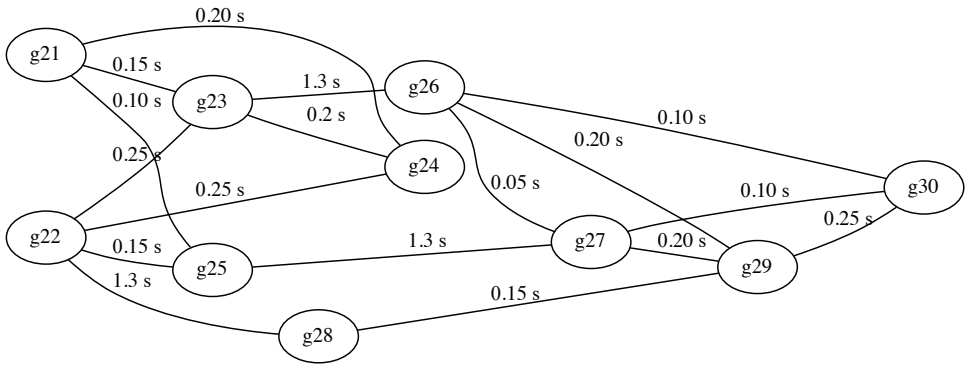

Figure 1. Topology of the network testbed used for scenario 2. Flow source: g21. Two-member anycast group: {g27, g30}.

The CSG software [16], which is in Python, was extended to support anycast routing. The gateway was configured with the Licklider transmission protocol (LTP) as the convergence layer (CL) protocol. LTP is a delay-tolerant protocol that can reliably deliver data blocks over an unreliable link with disruptions and long propagation delay. The main features of LTP were implemented: delay tolerance, block segmentation, reliable block delivery via multi-round transmissions, and the use of parallel transmission sessions (configured as 32 sessions). The latter feature allows the protocol to achieve high efficiency over large delay-bandwidth links. The same CL was used for all of the tested scenarios to obtain

comparable results. For performance reference, the CGR algorithm was also implemented. *CGR is the standard routing method used by Schedule-Aware Bundle Routing*, and so it provides an adequate reference of the performance level of the current state-of-practice of DTN. The parameters used by the CNC are shown in Table 1. Each experiment consisted of observing the bundle delivery performance of a flow addressed to the anycast set {g27, g30}, and sent from node g21. The flow included 100 kB bundles, which were sent at a designated rate. The transmission was stopped after 1000 bundles. Since a reliable convergence layer was used, the final experiment statistics were collected after all of the bundles arrived at the anycast destination. Note that the intention is not to find the steady-state results, but to characterize the average performance of a selected number of bundle transmissions that represent as a whole, for example, the transmission of a large scientific dataset split into 1000 bundles. The experiments were repeated multiple times and where possible; the 95% confidence interval is shown.

Table 1. Summary of the CNC parameters used in the tests.

Parameter	Value
SNN simulation time, step	100 ms
SNN simulation step	0.5 ms
Time constant (τ)	10 ms
Resistance (R)	10 MΩ
Maximum depolarization (spike voltage)	35 mV
Spike fire threshold (θ)	−50 mV
Reset	−90 mV
Rest	−70 mV
Bootstrap current (I_{ref})	2.1 nA
Random walk probability (P_w)	0.1
Learning factor (η)	0.1
Smoothing factor (α)	0.1

5. Performance Measurements

Mobile networks for space, air, ground, and sea applications involve the use of channels of different characteristics that are affected by particular environmental conditions. To cover a wide range of cases while still allowing reasonable comparisons among the different cases, the evaluation study focused on a common network topology (depicted in Figure 1). Different cases were then evaluated by applying different conditions to the channel features and operating conditions.

5.1. Scenario 1: Homogeneous Propagation Delays and Random Packet Drops

The case of links sharing identical features was evaluated first. In this scenario, the propagation delay values that are depicted by the graph's edges in Figure 1 were replaced with a constant value of 100 ms. These delays were emulated by introducing controlled buffering to the outbound links of the nodes through the Traffic Control tool (NetEm). Three of the links were configured to randomly drop packets with a small probability: 0.01 (g22–g28), 0.02 (g23–g26), and 0.03 (g25–g27). These links emulate the use of wireless channels in the topology (e.g., via medium-orbit satellites). Figure 2a depicts the average response time achieved by the proposed anycast routing method. For comparison purposes, the results obtained with CGR (unicast as the protocol does not support anycast) to either of the anycast group members are also shown. Addressing the whole traffic with CGR to g27 rather than g30 produced better results. The performance achieved by the CSG method was observed to be better than either choice sent with CGR. CSG dynamically divided the flow delivering a number of the bundles to g27 and the rest to g30 as the SNN adapted to the expected routing cost. This effect can be observed in Figure 2b, which depicts the average path of the flows and sub-flows. The average observed split between the two sub-flows is depicted in Figure 2d. It is interesting to note that under low traffic levels, CSG attempted to balance in about the same proportion the workload of both anycast group

members. However, as the traffic level was increased, there was a clearer preference for the g27 node because of the larger end-to-end bandwidth to that node. The response time deviation of the resulting sub-flow from the aggregate average was observed to be very small in this experiment. The average response time of the sub-flows is depicted by the discontinuous lines in the figure. The addition of both sub-flow throughputs yielded the final CSG throughput.

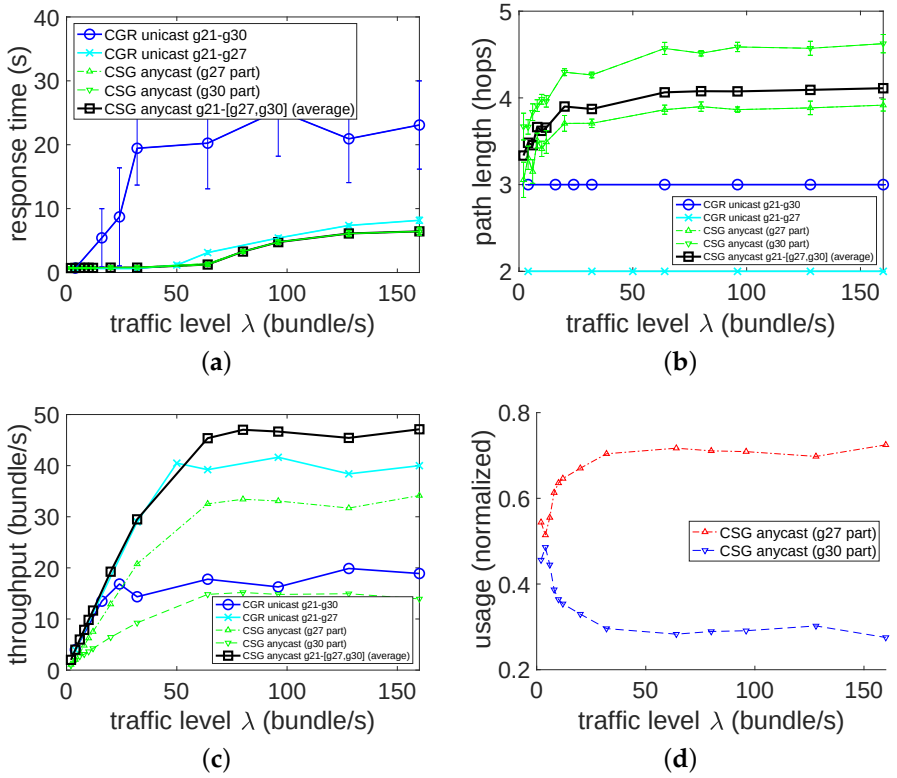

Figure 2. Scenario 1: Homogeneous propagation delays and random packet drops. (**a**) Average response time; (**b**) average path length; (**c**) bundle throughput; (**d**) traffic split.

Lower bundle response times imply a more balanced use of the parallel paths that are available for forwarding bundles in the DTN. Both routing methods use this metric as the minimization goal. By reducing the response times, it is also possible to improve the end-to-end throughput as indicated by the results in Figure 2c compared to what can be achieved with unicast CGR to a selected member.

5.2. Scenario 2: Earth–Moon Scenario

Mobile networks of nodes in the space, air, ground, and sea domains may include links of large one-way light times. To evaluate one such case, a possible earth–lunar network was considered by modifying the propagation delays of the testbed. In this case, data need to be downloaded from a node in the lunar section to any of the two designated ground stations. Three links were configured as long-haul connections: g23–g26, g25–g27, and g22–g28, with a 1.3 s propagation delay for each communication direction. No random packet losses were introduced in this test. The propagation delays within each of the sections were selected to be in the range of 7–150 ms to represent different ground, satellite–satellite, and satellite–ground links, with satellites at MEO and LEO orbits.

The link propagation delays that were in general longer that in the previous scenario directly impacted the response times of the flow. This effect is depicted in Figure 3a, where the average response times of either of the unicast routing transmissions were up to 10 times larger than the one obtained with the anycast CGR once congestion started forming. As with the first experiment, the lower response times were achieved by exploiting longer but

less congested paths, as shown in Figure 3b. The g21–g30 sub-flow that was dynamically determined by CSG was transmitted over a longer path on average than the other sub-flow. As with the first scenario, a larger throughput was observed for medium to high traffic levels, as shown in Figure 3c. The average observed traffic split between the two group members as dynamically determined by the CSG anycast routing is depicted in Figure 3d with the g27 target achieving a slightly higher preference over the g30 target.

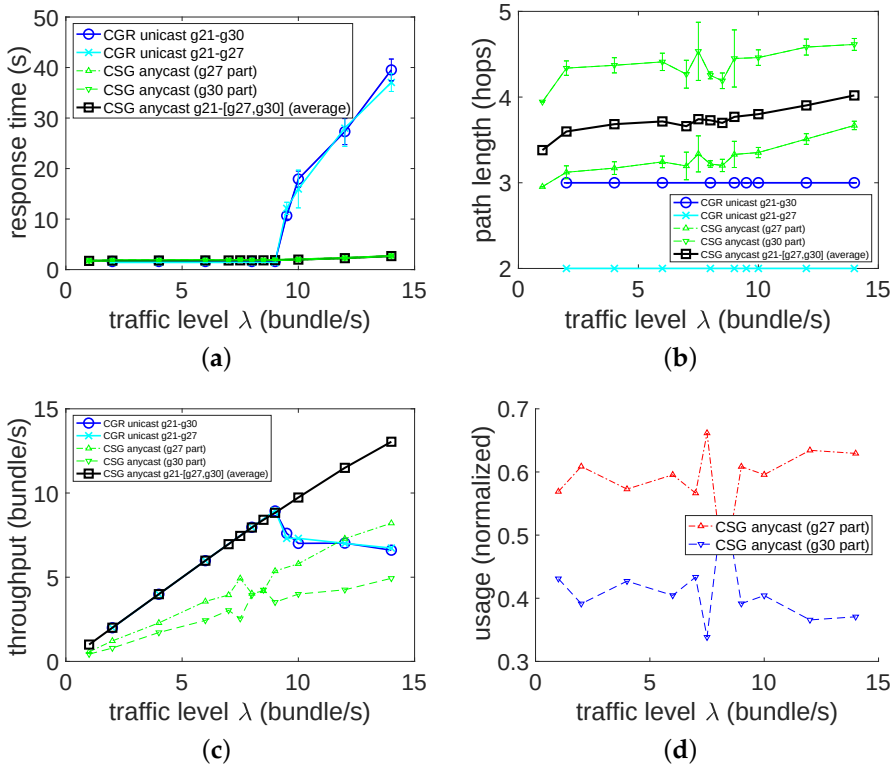

Figure 3. Scenario 2: Emulated earth–moon network topology. (**a**) Average response time; (**b**) average path length; (**c**) bundle throughput; (**d**) traffic split.

5.3. Scenario 3: Regular Link Disruptions

Link disruptions can occur in mobile networks for different reasons in addition to regular link and node failures. It is well known that random graphs show a zero-one phase transition for network connectivity; thus, if the node density drops below the critical threshold, the chances of a network partition greatly increase [44]. Environmental factors, such as obstacles, can also cause link disruptions. Satellite links in the LEO, MEO, or HEO can experience loss of visibility for several minutes or hours and even longer periods beyond the cislunar range.

Repetitive disruption patterns were dynamically applied to two of the long-haul links of the network as follows. Link g23–g26 was kept active for 11 s and then disabled for 5 s. The active–inactive time lengths for Link g25–g27 were chosen to be 7 s and 13 s, respectively. Given that the experiments were run in real-time, the contact and disruption times were chosen to be small to achieve manageable testing times. The disruption patterns were controlled at runtime by the NEE, which, as explained earlier, runs as a distributed process parallel to the CSG.

The link disruptions reduce the end-to-end carrying capacity of the network, which the CSG attempts to dynamically discover and exploit. For the given topology and disruption pattern, this task is achieved by using longer paths, as shown in Figure 4b. The penalty of this capacity reduction can be observed through the longer average response times of the flow bundles, as depicted in Figure 4a. Unlike the previous case, the disruptions accentuated the performance differences between the two anycast group members, with

unicast routing to g30 offering greater bandwidth than to g27. It is interesting to observe that the CGR actually preferred g27 in larger proportion to g30 as the delivery target as it lay along the faster path, as shown in Figure 4d. These observations also apply to the flow throughput that is depicted in Figure 4c.

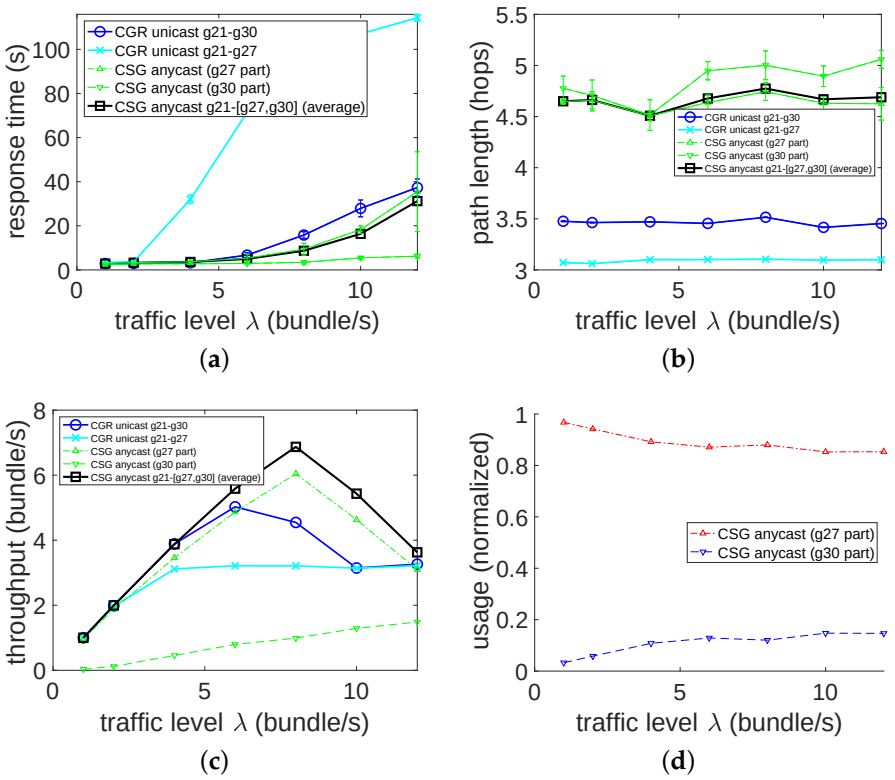

Figure 4. Scenario 3: Network with regular link disruptions. (**a**) Average response time; (**b**) average path length; (**c**) bundle throughput; (**d**) traffic split.

5.4. Anycast Group Size

Although it would depend on the layout of the network topology and the group member that are chosen, the general expectation is that larger anycast groups would involve shorter delivery times. To verify this observation, a set of experiments was designed with different group sizes. Starting from the two-member case of the prior sections, new group members were added in the following order: g29, g26, g28, g22, g24, g23, and g25, yielding group sizes from 2 to 9. To define this sequence, the farthest node from the source was selected to join the anycast group for the next experiment.

Figure 5 depicts the average response time as a function of the anycast group size. Four cases are shown, which correspond to traffic levels of 10, 20, 50, and 100 bundles. The expected response times become larger with higher traffic levels, but smaller with larger groups. A significant change can be observed in the response time after increasing the group size to 6. One of the reasons for this outcome is that all of the first five gateways in the group belong the ground section of the network and so all feasible paths must cross one of the long-haul links. The first gateway located at the space section side, where the flow source is also located, is g22, which was added to make a group of size 6.

Another reason for the large change in the response time after adding node g22 is because of the network topology, whose structure allows a significant reduction in the path length once the group size becomes at least 6 (see Figure 6). Additionally, note that the differences on the average path length tend to diminish as the anycast group becomes larger due to the reduced path diversity.

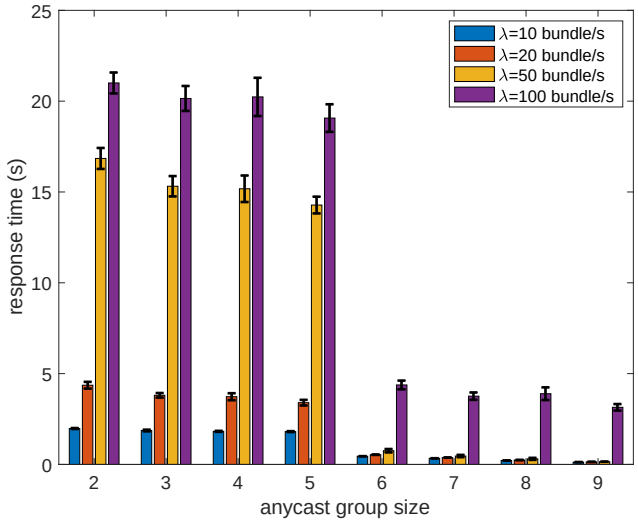

Figure 5. Effect of the anycast group size on the average response time of the flow. Given the vector M = [g29, g26, g28, g22, g24, g23, g25], the anycast group A of size n, $n = 1, \ldots, |M|$ was defined as $A = \{M_i\}, i = 1, \ldots n$.

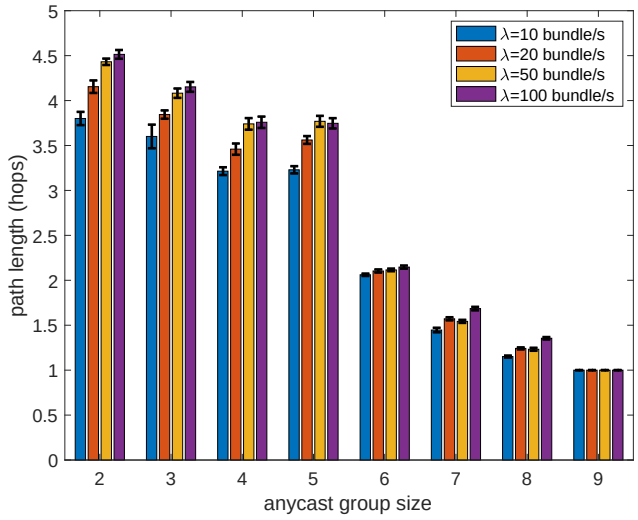

Figure 6. Smaller path lengths were observed on average with larger anycast group sizes.

The observations regarding the average flow throughput follow a similar logic with one difference (see Figure 7). For light traffic levels (i.e., 10 or 20 bundles) the throughput remained almost constant regardless of the group size. This happened because no saturation was reached with those traffic levels and thus the observed throughput was basically identical to the flow rate. With large traffic levels, congestion is created in parts of the network and the throughput is less than the flow rate. For those cases, the addition of space-section gateways to the group (size 6 onwards) allowed boosting the flow throughput significantly, thus removing all congestion at least for the case of $\lambda = 50$.

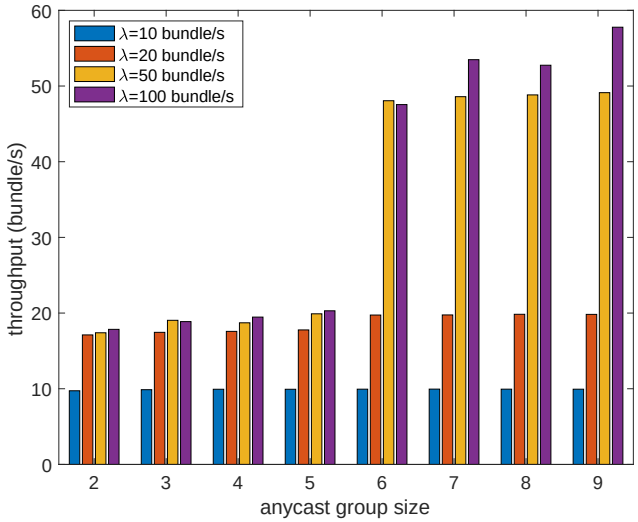

Figure 7. Throughput as a function of the group size.

6. Conclusions

By taking advantage of the autonomous route decision-making capability of the cognitive space gateway method, an anycast routing mechanism for delay-tolerant networks was developed. It allows network gateways to independently determine the optimal outbound link to be used that leads to the lowest delivery time to one of the members of the anycast group. Measurements obtained with a system prototype have shown significant performance improvement over unicasting given that anycasting is not currently supported by the standard space DTN approach (i.e., the CGR algorithm). Future work is expected to further evaluate the approach using representative traffic of actual applications.

Funding: This research was funded by the NASA Space Technology Research Grants Program #80NSSC17K0525.

Conflicts of Interest: The author declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BER	Bit Error Rate
CGR	Contact Graph Routing
CL	Convergence Layer
CCSDS	Consultative Committee for Space Data Systems
CNC	Cognitive Network Controller
CSG	Cognitive Space Gateway
DTN	Delay-Tolerant Networking
LIF	Leaky-Integrate-and-Fire
LTP	Licklider Transmission Protocol
SNN	Spiking Neural Network
UTP	Unshielded Twisted Pair

References

1. Fu, Q.; Rutter, B.; Li, H.; Zhang, P.; Hu, C.; Pan, T.; Huang, Z.; Hou, Y. Taming the Wild: A Scalable Anycast-Based CDN Architecture (T-SAC). *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2757–2774. [CrossRef]
2. Xue, J.; Dang, W.; Wang, H.; Wang, J.; Wang, H. Evaluating Performance and Inefficient Routing of an Anycast CDN. In Proceedings of the 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), Phoenix, AZ, USA, 24–25 June 2019; pp. 1–10.
3. Bakiras, S. Approximate server selection algorithms in content distribution networks. In Proceedings of the 2005 IEEE International Conference on Communications (ICC 2005), Seoul, Korea, 16–20 May 2005; Volume 3, pp. 1490–1494. [CrossRef]

4. Tran, H.A.; Mellouk, A.; Hoceini, S.; Perez, J. User-centric Content Distribution Network architecture. In Proceedings of the 2012 Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), St. Petersburg, Russia, 3–5 October 2012; pp. 343–350
5. de Vries, W.B.; Schmidt, R.d.O.; Pras, A. Anycast and Its Potential for DDoS Mitigation. In *Management and Security in the Age of Hyperconnectivity*; Badonnel, R., Koch, R., Pras, A., Drašar, M., Stiller, B., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 147–151.
6. Taghizadeh, S.; Elbiaze, H.; Bobarshad, H. EM-RPL: Enhanced RPL for Multigateway Internet-of-Things Environments. *IEEE Internet Things J.* **2021**, *8*, 8474–8487. [CrossRef]
7. Davis, F.; Marquart, J.; Israel, D.J. A DTN-Based Multiple Access Fast Forward Service for the NASA Space Network. In Proceedings of the 2011 IEEE Fourth International Conference on Space Mission Challenges for Information Technology, Palo Alto, CA, USA, 2–4 August 2011; pp. 64–65. [CrossRef]
8. Lent, R. An Anycast Service with Cognitive DTN Routing. In Proceedings of the 2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), New Delhi, India, 14–17 December 2020; pp. 1–6. [CrossRef]
9. Pachler, N.; del Portillo, I.; Crawley, E.F.; Cameron, B.G. An Updated Comparison of Four Low Earth Orbit Satellite Constellation Systems to Provide Global Broadband. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–7. [CrossRef]
10. Coutinho, R.W.L.; Boukerche, A. North Atlantic Right Whales Preservation: A New Challenge for Internet of Underwater Things and Smart Ocean-Based Systems. *IEEE Instrum. Meas. Mag.* **2021**, *24*, 61–67. [CrossRef]
11. Usman, O.B.; Knopp, A. Digital Predistortion in High Throughput Satellites: Architectures and Performance. *IEEE Access* **2021**, *9*, 42291–42304. [CrossRef]
12. Ahmed, T.; Ferrus, R.; Fedrizzi, R.; Sallent, O.; Kuhn, N.; Dubois, E.; Gelard, P. Satellite Gateway Diversity in SDN/NFV-enabled satellite ground segment systems. In Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 21–25 May 2017; pp. 882–887.
13. Quibus, L.; Le Mire, V.; Queyrel, J.; Castanet, L.; Féral, L. Rain Attenuation Estimation with the Numerical Weather Prediction Model WRF: Impact of Rain Drop Size Distribution for a Temperate Climate. In Proceedings of the 2021 15th European Conference on Antennas and Propagation (EuCAP), Dusseldorf, Germany, 22–26 March 2021; pp. 1–5. [CrossRef]
14. Burleigh, S.; Hooke, A.; Torgerson, L.; Fall, K.; Cerf, V.; Durst, B.; Scott, K.; Weiss, H. Delay-tolerant networking: An approach to interplanetary Internet. *IEEE Commun. Mag.* **2003**, *41*, 128–136. [CrossRef]
15. Araniti, G.; Bezirgiannidis, N.; Birrane, E.; Bisio, I.; Burleigh, S.; Caini, C.; Feldmann, M.; Marchese, M.; Segui, J.; Suzuki, K. Contact graph routing in DTN space networks: Overview, enhancements and performance. *IEEE Commun. Mag.* **2015**, *53*, 38–46. [CrossRef]
16. Lent, R. A Neuromorphic Architecture for Disruption Tolerant Networks. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
17. Lent, R.; Brooks, D.; Clark, G. Validating the Cognitive Network Controller on NASA's SCaN Testbed. In Proceedings of the 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 554–559.
18. Lent, R. A Cognitive Network Controller Based on Spiking Neurons. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
19. Lines, A.; Joshi, P.; Liu, R.; McCoy, S.; Tse, J.; Weng, Y.; Davies, M. Loihi Asynchronous Neuromorphic Research Chip. In Proceedings of the 2018 24th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), Vienna, Austria, 13–16 May 2018; pp. 32–33.
20. Sommese, R.; Bertholdo, L.; Akiwate, G.; Jonker, M.; van Rijswijk-Deij, R.; Dainotti, A.; Claffy, K.; Sperotto, A. MAnycast2: Using Anycast to Measure Anycast. In Proceedings of the ACM Internet Measurement Conference (IMC '20), New York, NY, USA, 27–29 October 2020; pp. 456–463. [CrossRef]
21. Yang, Y.; Shi, X.; Yin, X.; Wang, Z.; Zhou, X. The Understanding and Forecast of AS-Level Anycast Path Inflation. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; pp. 1–7. [CrossRef]
22. Bertholdo, L.M.; Ceron, J.M.; Granville, L.Z.; Moura, G.C.M.; Hesselman, C.; van Rijswijk-Deij, R. BGP Anycast Tuner: Intuitive Route Management for Anycast Services. In Proceedings of the 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2–6 November 2020; pp. 1–7. [CrossRef]
23. McQuistin, S.; Uppu, S.P.; Flores, M. Taming Anycast in the Wild Internet. In Proceedings of the Internet Measurement Conference (IMC '19), New York, NY, USA, 21–23 October 2019; pp. 165–178. [CrossRef]
24. Hao, F.; Zegura, E.W.; Ammar, M.H. QoS routing for anycast communications: Motivation and an architecture for DiffServ networks. *IEEE Commun. Mag.* **2002**, *40*, 48–56.
25. Zaumen, W.T.; Vutukury, S.; Garcia-Luna-Aceves, J.J. Load-balanced anycast routing in computer networks. In Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000), Antibes-Juan Les Pins, France, 3–6 July 2000; pp. 566–574.
26. Bathula, B.G.; Vokkarane, V.M.; Lai, C.P.; Bergman, K. Load-Aware Anycast Routing in IP-over-WDM Networks. In Proceedings of the 2011 IEEE International Conference on Communications (ICC), Kyoto, Japan, 5–9 June 2011; pp. 1–6.

27. Li, Y.; Han, Z.; Gu, S.; Zhuang, G.; Li, F. Dyncast: Use Dynamic Anycast to Facilitate Service Semantics Embedded in IP address. In Proceedings of the 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), Paris, France, 7–10 June 2021; pp. 1–8. [CrossRef]
28. Zegura, E.W.; Ammar, M.H.; Fei, Z.; Bhattacharjee, S. Application-layer anycasting: A server selection architecture and use in a replicated Web service. *IEEE/ACM Trans. Netw.* **2000**, *8*, 455–466. [CrossRef]
29. Cao, Y.; Wang, T.; Zhang, X.; Kaiwartya, O.; Eiza, M.H.; Putrus, G. Toward Anycasting-Driven Reservation System for Electric Vehicle Battery Switch Service. *IEEE Syst. J.* **2019**, *13*, 906–917. [CrossRef]
30. Velusamy, G.; Lent, R. An Adaptive Approach for Demand-Response and Latency Control in Distributed Web Services. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6. [CrossRef]
31. Kim, J.; Lin, X.; Shroff, N.B.; Sinha, P. Minimizing Delay and Maximizing Lifetime for Wireless Sensor Networks With Anycast. *IEEE/ACM Trans. Netw.* **2010**, *18*, 515–528.
32. Zhao, Z.; Min, G.; Dong, W.; Liu, X.; Gao, W.; Gu, T.; Yang, M. Exploiting Link Diversity for Performance-Aware and Repeatable Simulation in Low-Power Wireless Networks. *IEEE/ACM Trans. Netw.* **2020**, *28*, 2545–2558. [CrossRef]
33. Fall, K. A Delay-tolerant Network Architecture for Challenged Internets. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA, 25–29 August 2003; pp. 27–34. [CrossRef]
34. Tuan Le.; Gerla, M. Social-Distance based anycast routing in Delay Tolerant Networks. In Proceedings of the 2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Vilanova i la Geltru, Spain, 20–22 June 2016; pp. 1–7.
35. Le, T.; Gerla, M. An anycast routing strategy with time constraint in delay tolerant networks. In Proceedings of the 2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Budva, Montenegro, 28–30 June 2017; pp. 1–6.
36. Net, M.S.; Burleigh, S. Evaluation of Opportunistic Contact Graph Routing in Random Mobility Environments. In Proceedings of the 2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), Huntsville, AL, USA, 11–13 December 2018; pp. 183–188.
37. Zeng, D.; Teng, C.; Yao, H.; Liang, Q.; Hu, C.; Yan, X. Stochastic Analysis of Epidemic Routing Based Anycast in Throwbox-Equipped DTNs. In Proceedings of the 2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs, Aizu-Wakamatsu, Japan, 23–25 September 2014; pp. 77–81.
38. Gong, Y.; Xiong, Y.; Zhang, Q.; Zhang, Z.; Wang, W.; Xu, Z. WSN12-3: Anycast Routing in Delay Tolerant Networks. In Proceedings of the IEEE Globecom 2006, San Francisco, CA, USA, 27 November–1 December 2006; pp. 1–5.
39. Hadi, F.; Shah, N.; Syed, A.H.; Yasin, M. Effect of Group Size on Anycasting with Receiver Base Forwarding in Delay Tolerant Networks. In Proceedings of the 2007 International Conference on Electrical Engineering, Lahore, Pakistan, 11–12 April 2007; pp. 1–4.
40. Rosa da Silva, E.; Guardieiro, P.R. Anycast routing in Delay Tolerant Networks using genetic algorithms for route decision. In Proceedings of the 2008 11th International Conference on Computer and Information Technology, Khulna, Bangladesh, 24–27 December 2008; pp. 65–71.
41. Gerstner, W.; Kistler, W. *Spiking Neuron Models: An Introduction*; Cambridge University Press: New York, NY, USA, 2002.
42. Lent, R. Evaluation of Cognitive Routing for the Interplanetary Internet. In Proceedings of the 2020 IEEE Global Communications Conference (GLOBECOM 2020), Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [CrossRef]
43. ION-DTN. The Interplanetary Overlay Network (ION) Software Distribution. Available online: https://sourceforge.net/projects/ion-dtn (accessed on 27 April 2021).
44. Krishnamachari, B.; Wicker, S.B.; Béjar, R.; Pearlman, M. Critical Density Thresholds in Distributed Wireless Networks. In *Communications, Information and Network Security*; Springer: Boston, MA, USA, 2003; pp. 279–296. [CrossRef]