DTIPN: Delay Tolerant IP Networking for Opportunistic Network Applications

Hideya Ochiai
The University of Tokyo/NICT
jo2lxq@hongo.wide.ad.jp

Kenichi Shimotada
The University of Tokyo
ken@hongo.wide.ad.jp

Hiroshi Esaki
The University of Tokyo/NICT
hiroshi@wide.ad.jp

ABSTRACT

This paper proposes delay tolerant IP networking (DTIPN), an alternative architecture for delay and disruption tolerant networks. This architecture directly fits into the Internet protocol architecture, still providing delay and disruption tolerant support for application message delivery without relying on the well-known Bundling. DTIPN takes IP packets as asynchronous data delivery units, identifying the location of hosts by IP addresses. We have implemented the architecture and carried out several experiments with 10 intermittently-connected or mobile nodes in the campus of the University of Tokyo. The result, on our prototype implementation, indicates that DTIPN architecture has well-adaptive nature to the Internet with providing practically useful performance.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Network Architecture and Design—*Store and forward networks*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol architecture*

General Terms

Design, Experiments, Performance

Keywords

Delay Tolerant Networks, Internet Architecture, Campus-Wide Experiment

1. INTRODUCTION

In the challenged network environment, any communication nodes are always virtually networked over intermittent physical connection. This idea opens up the new communication paradigm that extends the communication infrastructure everywhere from tiny embedded devices to even out of the Earth.

The impact of the emergence of delay tolerant networks (DTN)[8, 4, 5] is remarkable. Sensor networking community is now applying the concept to its framework[18, 21]. Mobile ad-hoc networks (MANET) reseach community is also tring to use it[17, 1]. Researches of vehicular ad-hoc networks (VANET) cannot be discussed without DTN[6, 12].

However, the widely-discussed DTN, which was originally designed for interplanetary communication, is still one of the approaches for the challenged network environment. IP-based sensor networking or MANETs with 100-node scale can take another approach with proposing alternative architecture for delay tolerant networking. The architecture of widely-discussed DTN is not well-adaptive to the Internet, and the management and operational issues have not likely been deeply discussed, which is mandatorily required in the deployment phase.

This paper proposes delay tolerant IP networking (DTIPN) architecture, which we have developed as a different style of delay tolerant networking, assuming about 100 nodes in a challenged network segment. The architecture does not rely on *Bundling*, which has characterized the well-known DTN. It uses IP address for node locators, still providing delay tolerant support for application message delivery. DTIPN is much more adaptive to the existing IP network than the widely-discussed DTN, regarding to practical management and network operation.

Generally, all the communication protocols targeted at the challenged network environment must avoid synchronous communication style. Making state synchronization over physically disrupted connectivity is basically impossible or it would get only a weak synchronization. The DTN has avoided synchronous communication between communication ends and enabled asynchronous message delivery, by dividing an end-to-end TCP session into multiple sub-sessions.

DTIPN, our approach, takes IP packets as asynchronous data delivery units. IP networks originally provide asynchronous packets delivery, and it is basically allowed for large delay: even one hour or one day delay. In DTIPN, the data link layer for the challenged environment enables IP packet delivery over physical intermittent connectivity, and the transport (or, the application) protocol layer supports the long-delay delivery of application protocol data units (APDUs).

DTIPN is much more adaptive to the Internet architecture. Anyone can easily deploy independently around their Internet edges with the conventional management operation. The widely-discussed DTN requires new management and operational schemes, especially when we come to deploy it

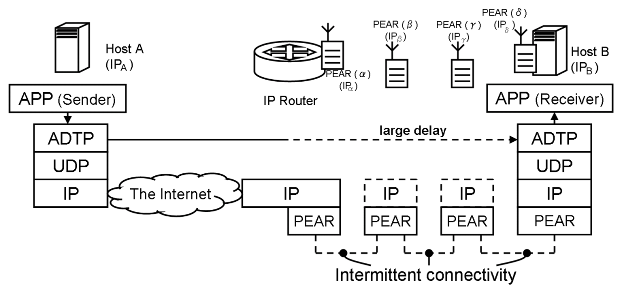

Figure 1: Delay tolerant IP networking architecture

as a globally managed network, because it basically establishes a new network in the overlay manner.

One of the contributions of this paper is to demonstrate how the architecture can be implemented and how it is practically useful. For this purpose, we have implemented DTIPN and carried out some experiments, and we present the performance of our prototype system in this paper. At the data link layer, we adopted potential-based entropy adaptive routing (PEAR), which was proposed in our previous work[15], as an IP packet delivery framework for wide-range of mobility models. At the application protocol layer, we implemented a forward error correction (FEC) scheme, which would be practically useful for moderate packet loss in the Internet space. Thus, the evaluation work is totally based on the real implementation. We have carried out four types of experiments with 10 nodes using the campus of the University of Tokyo.

This paper is organized as follows. Section 2 provides the related work. We propose DTIPN architecture in section 3. Section 4 provides our evaluation work with presenting the prototype implementation. Section 5 gives the discussion, and finally this paper provides the summary in section 6.

2. RELATED WORK

The widely-discussed DTN was proposed by Burleigh et. al. [4] and Kevin Fall[8], and the internet engineering task force(IETF) has published it as RFC4838[5]. The architecture is characterized by the Bundle layer, which deploys DTN framework as an overlay network and enables asynchronous message delivery. The background of the architecture came from the study of TCP performance reduction and failure over large delay and high packet loss networks[11]. In order to avoid TCP-based synchronous communication between communication ends, they have proposed to take hop-by-hop delivery at the Bundle layer.

DTIPN takes different architecture to enable asynchronous message delivery over the challenged environment. It uses IP packets as asynchronous data delivery units, and enables application message delivery by the transport or the application layers. This approach directly fits into the Internet protocol architecture, indicating that DTIPN has great adaptivity to the current Internet.

This work inherits our previous works. IP over DTN[16] has proposed the basic concept for DTIPN. Potential-based entropy adaptive routing (PEAR)[15] is an autonomous message routing algorithm over the intermittently connected networks. One of the goals of this paper is to demonstrate the feasibility of deployment and the usability for practical applications.

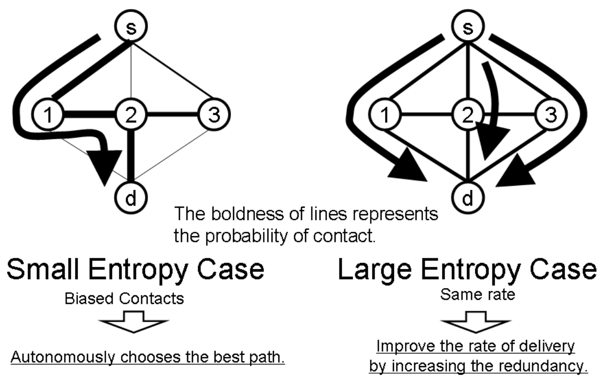
s
1
2
3
d
s
1
2
3
d
The boldness of lines represents the probability of contact.
Small Entropy Case
Biased Contacts
Autonomously chooses the best path.
Large Entropy Case
Same rate
Improve the rate of delivery by increasing the redundancy.

Figure 2: Entropy-adaptive message delivery

IP multicast practically takes non-reliable communication styles because of its difficulty in making state synchronization (i.e., data acknowledgement and retransmissions)[9], just as delay tolerant networking. In this regard, DTIPN can also take non-reliable communication styles. However, it would get reliability to a certain extent by forward error correction schemes just as Parity-based loss recovery[14] and Uni-DTN[10] has presented. We also take this approach at the transport (or, the application protocol) layer.

3. DELAY TOLERANT IP NETWORKING

This section describes the architecture for delay tolerant IP networking (DTIPN). It takes IP address as a communication endpoint identifier, uses IP packets as asynchronous data delivery units.

3.1 Requirements

We, here, summarize the basic requirements for DTIPN.

(1) A communication endpoint must be identified by an IP address. Even when the communication end is physically isolated, we consider that it is virtually connected to the network. Taking IP address for endpoint locator reduces the operational cost compared to the additionally required costs in the well-known DTN.

(2) The network must use an IP packet as an asynchronous data delivery unit. The delivery of an IP packet is one of the asynchronous communications. An IP network provides *send* and *recv* method, and it carries IP packets in the best effort manner. Basically, it is allowed to have large delay for packets delivery. Whether it delays one minute, one hour, or even one day, it does not matter if the delay is expected at the application level.

(3) The data link layer for the challenged network environment must assume possible link disruption and must save IP packets against possible losses as much as possible. We must design a framework that provides delay and disruption tolerant support for IP packet propagation over isolated network boundaries to deliver them even with large delay.

(4) The application layer protocol must not have shared states across a network. Interactive communication in a short time is no longer available in our target environment. Thus, all the application programs must assume asynchronous message delivery. The protocols used by those applications must not rely on synchronous communication style, too.

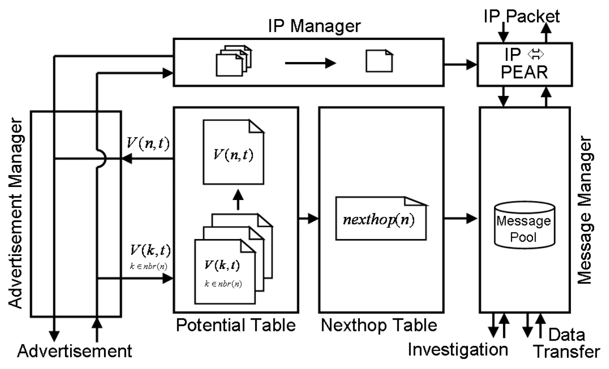

Figure 3: PEAR software design

3.2 Architecture

We present the architecture of DTIPN in figure 1. The architecture itself fits into the Internet protocol suite. The major features of this architecture are potential-based entropy adaptive routing (PEAR) at the data link layer and asynchronous data transfer protocol (ADTP) at the application protocol layer.

In fact, PEAR is one of the implementations of the link layer. Another message delivery scheme can be also applied here (e.g., [20, 13, 3, 7, 19]) as long as it provides delay and disruption tolerant support for IP packet delivery.

PEAR, in this architecture, autonomously enables the delivery of IP packets over intermittently connected networks in ad-hoc manner. Even if it takes a long time because of physical network disruption, it manages to deliver packets by its store-and-forward scheme. It adapts to wide-range of mobility models. Section 3.3 describes the internal design of PEAR.

ADTP enables asynchronous delivery of APDUs between remote application instances, making use of UDP for the under layer transport protocol. It provides *send* and *recv* methods as the application programming interface(API). ADTP must be tolerant for the delay of IP packet delivery – the receiver side should wait until it gets the whole APDU. ADTP does not have to provide reliable communication. Thus, some data loss should be assumed at the application programs. However, some efforts can be made in ADTP implementation to get reliability to a certain extent. Section 3.4 discusses in more detail.

The major difference with the widely-discussed DTN is that all the hosts have IP address for communication endpoint identifier. The widely-discussed DTN has its own identification schemes, and thus it should have its own network management and operational schemes. DTIPN directly fits into the existing Internet, which would help global delay tolerant networking with smaller operational costs.

3.3 PEAR for Link Layer Implementation

Potential-based entropy adaptive routing (PEAR) fits into the link layer in the DTIPN architecture. This section provides the overview and the software design of PEAR. For theoretical details (e.g., potentials and replica management scheme), please refer to our previous work[15].

3.3.1 Overview

PEAR, which was originally designed as a DTN framework, is practically useful for various scenarios. It performs effective routing over wide-range of mobility patterns in adhoc manner. A node learns what nodes are nearby and what nodes exist over intermittent connectivity. It performs routing and propagation of messages, adaptively changing the form of delivery pattern depending on the contact or mobility model.

Figure 2 shows how the message delivery pattern should change depending on the pattern of node contact. A bolder link indicates higher probability of node contact. This figure presents two networks; the left one is characterized as small entropy case, and the right one is as large entropy case. At the small entropy case, a node mostly meets with some particular nodes. As entropy increases, contact probability between any nodes becomes uniform. The definition of entropy is provided in section 4.3.

At small entropy cases, PEAR performs message delivery in the best form, by choosing the closer (and it is the best) hop at every transmission. However, at large entropy cases, which is no more deterministic and impossible to make any prediction, the main delivery force in PEAR is replication of messages in the network. This behavior is reasonable, because making replication increases the redundancy of the delivery path. Interestingly, PEAR achieves this adaptiveness without being aware of the mobility model itself.

As for transport, PEAR in DTIPN has only to deliver typical-sized IP packets. This enables system implementation quite simple (see, section 4.2).

3.3.2 Software Design

We provide a software design of PEAR in figure 3. It has six functional blocks. We, here, describe the details.

- **IP Manager:** IP Manager implements an epidemic-based IP-to-PEAR name map construction algorithm. This enables resolution of IP address to PEAR node name.

- **IP Encapsulator:** IP Encapsulator encapsulates an outgoing IP packet by a PEAR protocol frame, and decapsulates an incoming PEAR protocol frame.

- **Advertisement Manager(AM):** AM periodically advertises potentials to its neighbors by radio-range multicast. AM submits the received potentials to potential table (PT), and publishes the current potentials obtained from PT to the neighbors. AM also manages the dissemination of the map of IP address and PEAR node name.

- **Potential Table(PT):** PT manages all the potentials of neighbors and computes the potentials of the next time step by the potential-field construction algorithm[15].

- **Nexthop Table(NT):** NT generates and maintains nexthop table for each destination by the potentials managed at the PT.

- **Message Manager(MM):** MM implements the replica management scheme that PEAR defines[15]. It provides an interface of sending and receiving packets for IP Encapsulator.

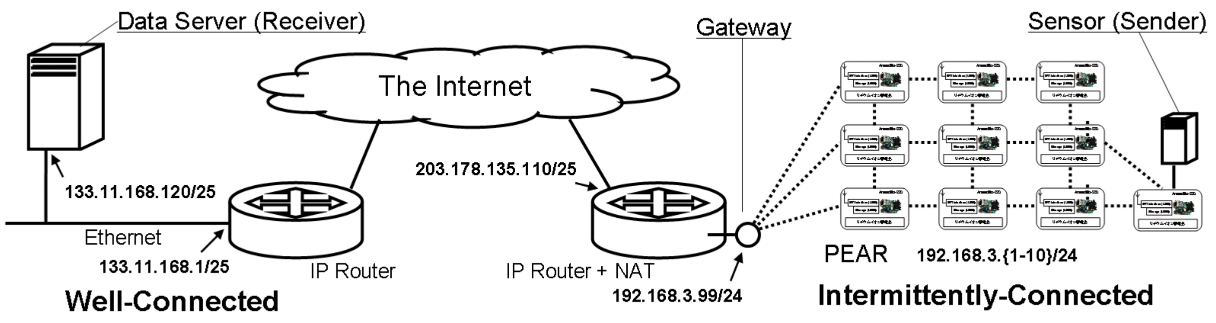

Figure 4: Network configuration for the experiment

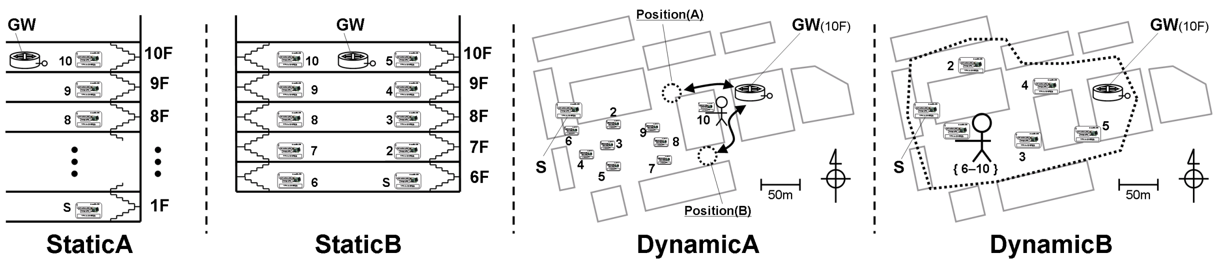

Figure 5: Experiment scenarios

3.4 Asynchronous Data Transfer Protocol

The role of ADTP is to provide the API for application instances to send and receive messages by application protocol data unit (APDU). It decomposes an APDU into UDP datagrams when sending it, and composes the original APDU from the received UDP datagrams. It must be tolerant to (1) moderate loss, (2) large latency, (3) out-of-order arrival and (4) duplicate arrival of IP packets. It should also be aware of traffic congestion; i.e., huge number of packets must not be sent in a burst manner.

PEAR carries IP packets at the link level, saving them from fatal losses as much as possible. However, even when it provides 100% packet delivery, the Internet sometimes drops them from the network. FEC, as widely discussed, must be the most practical method for the moderate data loss or error in asynchronous communication, and we also take this method for ADTP. However, we must also be aware that the packet loss might occur randomly or sometimes in burst. We can read this kind of discussion in the past literatures: e.g., parity-based reliable IP multicast transmission[14], and Uni-DTN[10].

We basically take reed-solomon algorithm as a FEC scheme. In order to implement it, assuming some burst losses and congestion, ADTP takes interleave and slow transmission.

4. EVALUATION

One of the contributions of this paper is to confirm the feasibility of deployment and to study the performance of the system with real prototype system in order to demonstrate the applicability to practical scenarios.

The first half of this section describes the settings of the experiment and the prototype implementation which we have developed. The latter part provides the analyses of performance and environmental features.

4.1 Experiment Settings

We have carried out experiments for both static-cases and dynamic-cases. We did on static cases, because even if nodes are statically setup, wireless links frequently disrupts in the real environment, causing intermittent connectivity in communication. This paper presents four types of experiment scenarios, which we call (1) StaticA, (2) StaticB, (3) DynamicA and (4) DynamicB.

Figure 4 is the basic network configuration applied to all the experiments. The data server in the well-connected network (133.11.168.0/25) receives messages from the sensor over the intermittently-connected network. The sensor periodically sends its messages to the server (133.11.168.120) by ADTP. PEAR network manages to deliver the IP packets to the gateway.

Figure 5 shows the physical deployment for each experiment. *S* is the message sender (i.e., sensor) and *GW* is the gateway for the upper Internet link.

StaticA: Nodes were statically deployed, one node for one floor in our building (i.e., Eng. Bldg.2, in the University of Tokyo).

StaticB: Nodes were statically deployed, two nodes for one floor in our building.

DynamicA: Nine nodes were statically deployed in the campus, and one node (No.10) has moved between Position(A) and GW, and Position(B) and GW alternately in 20 minutes per cycle.

DynamicB: Five nodes were statically deployed in the campus, and other five nodes have moved inside the dashed-area freely. They sometimes returned to the gateway (about three or four times) during the experiment.

For StaticA and StaticB, we conducted the experiment for 12 hours, and for DynamicA and DynamicB, the experiment was made for three hours and one hour respectively.

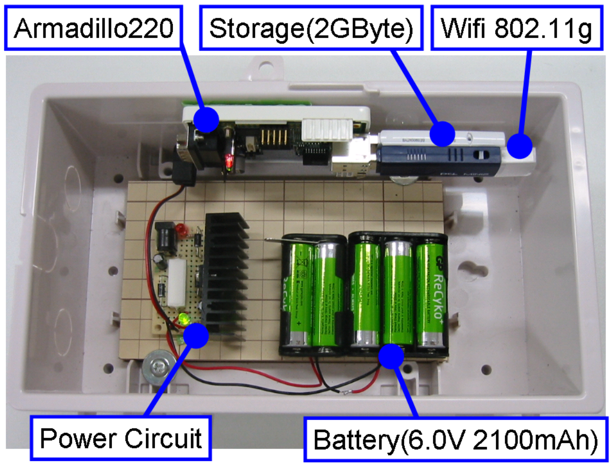

Figure 6: DTIPN prototype mobile node

During the experiment, S has sent 5k, 15k and 45k-byte dummy messages every 90 second each to the data server. According to the ADTP's FEC configuration, this generates 120 packets in 90 second: i.e., 80 [packet/min].

The parameter settings of PEAR were as follows: $D = 0.2$, $\rho = 0.02$, $\alpha = 0.04$, message TTL = 2400[sec]. The TTL of advertisement was 30 [sec], the time for dissemination was 1800 [sec], and the maximum buffer entry size was 4096. With this setting, maximum buffer occupancy would be around 3200 [entry]. Thus, it never exceeds the available buffer entry.

4.2 Prototype Implementation

We programmed in C and deployed into Armadillo-220[2]. Armadillo-220 is an embedded computer with 8Mbyte program memory and 32MByte working memory. It works with ARM9 200MHz CPU and Linux operating system. We here added an USB-stick IEEE802.11g module (GW-US54Mini2, Planex Communications Inc.) for ad-hoc communication with neighbors. We used linux-2.6.12.3-a9-15 for its kernel image. As for IPv6 support, please refer to section 5.

The footprint of PEAR is not large. The source code has only 3225 lines, and it works around 34k byte in object code size. We developed ADTP as a library for applications. The source code has 640 lines, including reed-solomon algorithm. The library has 32k byte in object code size.

We packed all of them into a handy box as figure 6. We assembled 11 boxes; one for the gateway and others for static or mobile nodes (including the sensor).

4.3 Contact Features

Figure 7 shows the aggregated contact graphs for each scenario. A line between two nodes shows contact. The bolder line indicates larger average contacted time. The presented average contact entropy was provided by the following definition.

$$S_c(n) = -\sum_{k\in(N-\{n\})} q_{e(n,k)} log(q_{e(n,k)}) \quad (1)$$

$$q_{e(n,k)} = \frac{p_{e(n,k)}}{\sum_{k\in(N-\{n\})} p_{e(n,k)}} \quad (2)$$

Here, $S_c(n)$ is the contact entropy of node n. $p_{e(n,k)}$ is the ratio of contact-time to the total-time between node n and k taken over statistically enough time period. N is the set of the nodes in a network.

As we can see, the contact graph certainly corresponds to the physical deployment; we can expect the physical location from these graphs. And we clearly found that even they are statically setup, the contact changes over time and the contacted ratios have varied.

We have also observed asymmetric contacts (e.g., node 2 received advertisement from node 3, but node 3 did not receive from node 2). However, it is not presented here because it makes the graph quite complicated.

4.4 Performance Analysis

4.4.1 *Potential-Field Construction*

Figure 8 shows the pattern of potential-field over time for StaticA, DynamicA and DynamicB. This shows the transition of node's potential-value associated to the gateway node. In drawing these sequences, we have omitted some nodes, for example, the DynamicA case only presents the potential of Sender, Node 7, Node 9 and Node 10.

In StaticA, potential-field is almost stable but with some ripples caused by intermittent connectivity even with neighbors. In DynamicA, node 10 always stayed with lower potential. In DynamicB, the potential-field became quite complicated with lots of upside-downs.

4.4.2 *Packet Delivery Pattern*

Figure 9 shows the pattern of IP packet delivery from the sender (S) to the gateway (GW). Each graph is associated to the delivery of one IP packet. The number along with an arrow shows the time when the IP packet has been copied; it is adjusted so that the GW arrival time is to be zero.

The delivery patterns have branches, making redundant paths for packet delivery. This sometimes has certainly improved delivery latency with increasing the overhead of transmissions and buffer occupancy.

We have also analyzed the average redundancy of delivery. We defined delivery redundancy R as $R = \frac{copyCount}{hopCount}$, where *copyCount* is the number of the same packets copied in the network, and *hopCount* is the number of hops from S to GW. If $R = 1$, no branches is made in delivering the packet. If it becomes large, lots of packet copies are made with improving delivery rate and delay. From the result, we can read that as entropy increased, PEAR has also increased the redundancy.

4.4.3 *Delivery Rate and Latency of IP Packets and APDUs*

Figure 10 shows the distribution of IP packet and APDU delivery latency for each experiment. The distribution was certainly dependent on the physical settings.

As we have expected, ADTP delivery has taken more time than IP packet delivery, though in DynamicB case, distribution of IP packet and APDU was almost the same. However, APDU was recovered by ADTP before receiving all the packets associated to the original message.

The packet delivery rate from S to GW was 100% for all the scenarios. The packet drop rate from GW to the data server was 0.50%; there were 10 hops in IP network from the GW to the server. ADTP has achieved 100% delivery.

4.4.4 *Control Plane Overheads*

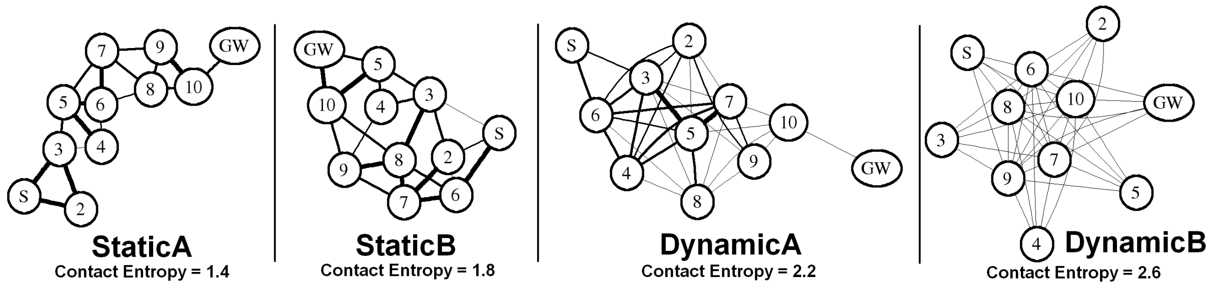

Figure 7: Summarized node contacts

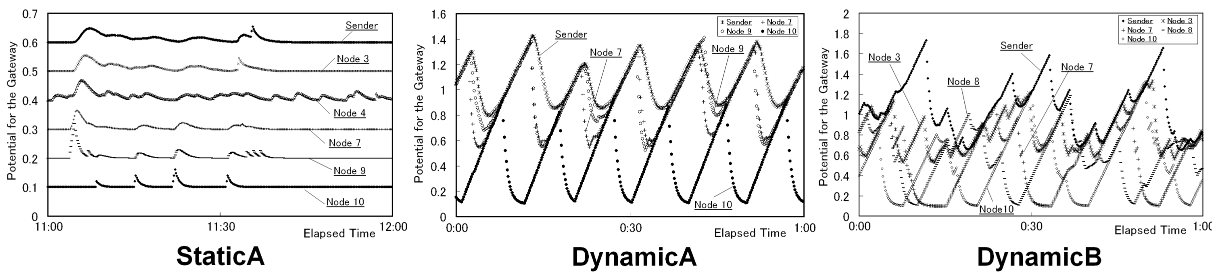

Figure 8: Potential-field changes over time

We have analyzed the ratio of control plane transmissions (i.e., advertisement and replica management) to the total transmissions in PEAR network. The result was approximately 10% – 20%, which indicates that transmission power was consumed mostly by data transfer rather than control signals.

We have also analyzed how the buffer entries were used. In StaticA and StaticB cases, more than 90% of the entries had cleared the data body, which indicates that most of the messages had been delivered in a short time and data-cleared entries remained there to delete the message from the network until the TTL expired. In DynamicA and DynamicB cases, about 30%–50% of the entries had data body. This occupancy acted for replicating packets in the network.

5. DISCUSSION

With our prototype-based experiment, we experienced that DTIPN would be practically useful for some opportunistic network applications. In our experiment settings, it has achieved 100% delivery of messages with acceptable overheads. This indicates that DTIPN is quite ready to be deployed in the real environment at the Internet edges.

Our prototype software also supports IPv6 already, though the experiment was carried out on IPv4 networks because the kernel version of the embedded computer was old, and the software could not use IPv6 properly. We confirmed that it worked on Linux kernel 2.6.28-15-generic (ubuntu 9.04), which means that the software itself is also ready for IPv6.

The architecture, especially the link layer, can be applied to any network configurations even to the core networks. However, we consider that it should be applied to edge networks in practice, and that it would be the major use case in the real situations.

DNS-based host identification and IP address resolution are not in the scope of this paper. IP address should be resolved beforehand or other additional methods should be explored to allow DNS-based networking.

6. CONCLUSION

We have proposed DTIPN, an alternative architecture for delay tolerant networking. DTIPN takes IP packets as asynchronous delivery units, identifying the location of hosts by IP addresses. In DTIPN, the data link layer provides disruption tolerant support in delivering IP packets over the challenged network environment, and the application protocol layer enables message transfer by APDU.

We have implemented the architecture with PEAR for the link layer and ADTP for the application protocol layer. We have carried out several experiments in the campus of the University of Tokyo, and analyzed the result regarding to delivery rate and latency, transmissions, and buffer occupancy. The result indicates that it is practically useful in the real environment.

The evaluation result was made on the real implementation of PEAR and ADTP. We confirmed, with these implementations, that DTIPN could adapt to the Internet very easily with practically useful performance.

7. REFERENCES

[1] G. F. Aggradi, F. Esposito, and I. Matta. Supporting predicate routing in DTN over MANET. In *ACM CHANTS*, 2008.

[2] Armadillo220. http://www.atmark-techno.com/.

[3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *IEEE INFOCOM*, 2006.

[4] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, jun 2003.

[5] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. RFC4838: delay tolerant networking architecture, apr 2007.

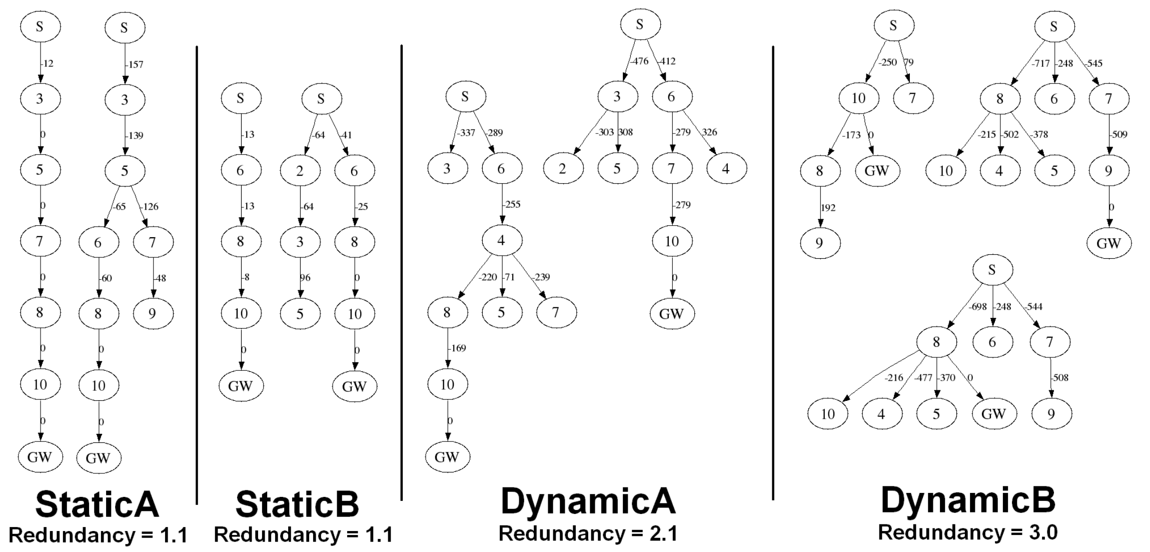

Figure 9: Packet delivery pattern

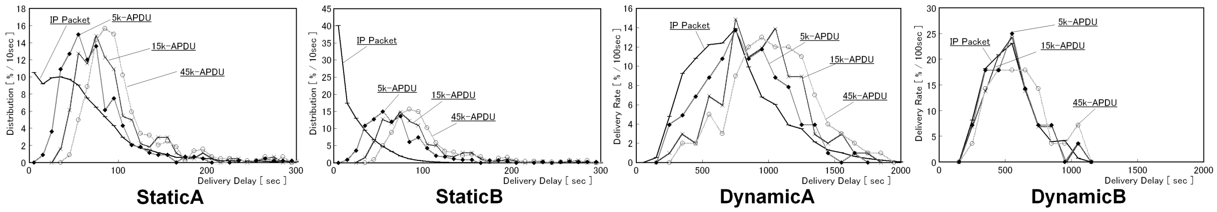

Figure 10: IP Packet and application message delivery latency

[6] P.-C. Cheng, K. C. Lee, M. Gerla, and J. Harri. GeoDTN+Nav: geographic dtn routing with navigator prediction for urban vehicular environments, jun 2009.

[7] D. Demmer and K. Fall. DTLSR: Delay tolerant routing for developing regions. In *ACM NSDR*, 2007.

[8] K. Fall. A delay-tolerant network architecture for challenged internets. In *ACM SIGCOMM*, 2003.

[9] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, dec 1997.

[10] D. Kutscher, J. Greifenberg, and K. Loos. Scalable dtn distribution over uni-directional links. In *ACM SIGCOMM Workshop*, 2007.

[11] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, jun 1997.

[12] F. Laurent and G.-C. Felipe. Using delay tolerant networks for car2car communications. In *IEEE Inductrial Electronics*, 2007.

[13] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *LNCS*, 3126:239–254, sep 2004.

[14] J. Nonnenmacher, E. W. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. *IEEE/ACM Transactions on Networking*, 6(4):349–361, aug 1998.

[15] H. Ochiai and H. Esaki. Mobility entropy and message routing in community-structured delay tolerant networks. In *ACM AINTEC*, pages 93–102, 2008.

[16] H. Ochiai, K. Shimotada, and H. Esaki. IP over DTN: large-delay asynchronous packet delivery in the Internet. In *IEEE ICUMT workshop*, 2009.

[17] J. Ott, D. Kutscher, and C. Dwertmann. Integrating DTN and MANET routing. In *ACM SIGCOMM workshop*, pages 221–228, 2006.

[18] B. Pasztor, M. Musolesi, and C. Mascolo. Opportunistic mobile sensor data collection with scar. In *IEEE MASS*, 2007.

[19] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: the multiple-copy case. *IEEE/ACM Transactions on Networking*, 16(1):77–90, feb 2008.

[20] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University, 2000.

[21] Y. Wang and H. Wu. Delay/fault-tolerant mobile sensor network (DFT-MSN): a new paradigm for parvasive information gathering. *IEEE Transactions on mobile computing*, 6(9), sep 2007.