TCPを用いたパケット再送を組み合わせた複数のIoTデバイスに対するマルチキャストによるファイル送信

五十嵐 蓮[1]　山本 拓海[1]　串田 高幸[1]

概要：通常，複数台の IoT デバイスに対するファイル送信は，PC と IoT デバイスをシリアルポートで接続し，一台ずつファイル送信を行う．しかしながら一台ずつ接続し，ファイル送信を行うとシリアルケーブルを繋ぎなおす処理に時間を費やしてしまう．したがって本稿では，UDP マルチキャストを用いた複数台の IoT デバイスに対するファイルの一斉送信を提案する．また，UDP のパケットロスによりファイル送信が失敗した IoT デバイスにはロスした特定のパケットを再送信する．再送信の際には，確実にパケットを届けるため TCP を用いることとする．評価として，本稿での提案と TCP シリアル送信のみの方法とで送信完了時間の比較を行った．実験は 100KiB と 500KiB のファイルサイズで場合分けを行い，その結果，受信機 5 台の場合において，TCP シリアル送信のみの方法が本提案より，100KiB では 2.85 秒，500KiB では 4.72 秒早いという結果となった．しかしながら本提案は台数の増加に関わらず，送信完了時間がほぼ一定だったため，受信機の台数とファイルサイズが増加するにつれ，本提案の方が早い送信が行えると予測される．

1. はじめに

背景

Internet of Things(以下: IoT) は，現代の生活において広く活用されており，インターネットを介して気象データや位置データといったセンサデータをやり取りしている [1]．日々，IoT デバイスの使用台数は増加している．これらの数は Safeatlast によると 2019 年には 266 億 6000 万台がアクティブだったと報告されており，2025 年までには 750 億台に上るとも予想されている [2]．ただし IoT ネットワークは，異なるデバイスや異なるリソース (CPU，メモリ，帯域) といった点で制約を抱えたデバイスが多数存在するため，普遍的なプロトコルでの通信が要求される [3]．通常, IoT デバイスへのファイル送信には PC と各 IoT デバイスを一台ずつシリアル接続しファイル送信を行う [4]．しかし，ファイル送信が必要な IoT デバイスが複数台存在し，それぞれにファイル送信を行う際には, シリアルケーブルを抜き差しするといった物理的な処理時間を要してしまう．それに加え，IoT デバイスは，人の手が届きにくい場所に設置する事も多く，シリアルケーブルの配線を行うこと自体コストがかかる行為である．そこで Transmission Control Protocol(以下: TCP) を用いたソケット通信や Hyper Text Transfer Protocol(HTTP) 通信を行うことにより，上述した様な物理的な配線処理を行う必要無くデータ送信が行える．しかし TCP や HTTP 通信を用いて複数の IoT デバイスにファイル送信を行う場合，1 対 1 での送信 (ユニキャスト) を行う必要があるため IoT デバイスの台数分，すべてのパケットを送る必要がある．そのため User Datagram Protocol(以下: UDP) を用いたファイル送信を提案する．

課題

本稿での課題として，送信側からのパケットを全て受信できたかの判断をどのタイミングで行うかの問題とパケットロスした特定のパケットをどのように判断するのかという主に 2 つが挙げられる．UDP は TCP と異なり，コネクションレス型で信頼性よりリアルタイム性を重視したプロトコルである．その特徴からファイル送信においてパケットロスが発生することを考慮しなくてはならない．そのため，受信側でパケットロス無く正常にパケットを受け取れたかの判断を行う必要がある．1 つ目の課題に関して，例えば，図 1 のように全ての送信パケット数が 10 個であると仮定した時に，パケットロスにより，1〜7 番目のパケットしか送られてこなかったとする．この場合，受信側は 10 個のパケットを受け取るよう認識しているため，残りの 3 個のパケットを待ち続けてしまう．よって受信側のファイル受信終了の判断が不可能である．最後の送信パケットに

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1

終了タグを付加する方法もあるが，UDP の特性により最後のパケットがロスした場合，判断不可能である．2 つ目に関しては，UDP のコネクションレス型の特徴によりパケット毎に受信可否の応答を行わない．そのため，受信完了後にパケットロスが判明してもどのパケットが脱落したか特定できない．

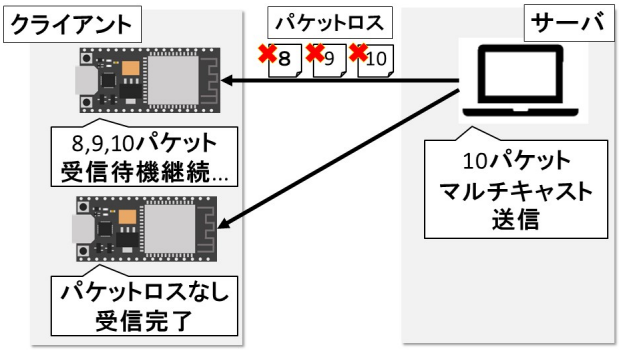

図 1　パケットロスを判断できず受信継続してしまうクライアント

各章の概要

第 2 章 関連研究では，関連研究について述べる．第 3 章 提案では，課題に対するアプローチを提案し，さらに具体的なユースケースについて言及する．第 4 章 実装では，提案を実現するために開発したソフトウェアとハードウェアの説明を行う．第 5 章 実験と分析では，本提案の有効性を検証する評価方法と評価結果を提示する．第 6 章 議論では，本稿での提案，実験，評価に関する議論すべき内容や改善案について述べる．第 7 章 まとめでは，本稿のまとめと今後の展望について述べる．

2. 関連研究

Dan Li らの研究では，オーバレイネットワークを利用することで受信機同士で同期を行い Peer to Peer でのパケットロス再送機能を実装する RDCM を提案している．しかしこれは，データセンタでの支障が想定されているため，リソースが限られた IoT ネットワークにおいいて，プロトコルのオーバーヘッドを許容できるか不明である [5].

Mahfoudhi, F らの研究では，Low Power Wide Area-network(以下: LPWA) 環境での Narrow band-IoT(以下: NB-IoT) を利用した Over-the-Air(OTA) ファームウェアのアップデートに関する研究を行っている．LPWA とは，既存の無線技術と比較して低電力で長い通信範囲を提供することで，多くの IoT 機器で構成されるネットワークを実装できる代表的な手段である [6]．この研究では NB-IoT と他の代表的プロトコルである LoRaWAN とのファイル送信に関する送信完了時間やパケット損失率，消費電力といった観点から比較を行っている．しかしこの手法は LPWA の環境に限ったものでありファームウェアの更新にも最大 4337.98 秒費やすことが示されており迅速なアップデートには適さない [7].

Geithner, T らの研究では，NACK-Oriented Reliable Multicast(以下: NORM) や NORM+ATRaCT，TCP シリアル送信，TCP 並列送信の 4 つの通信方式において伝送速度を評価おり，想定されたいくつかのケースに関しても NORM+ATRaCT の優位性を検証している．この研究では，送信機を中心として，受信機の配置を円形，らせん状，ランダム状に配置し，それぞれの場合とで伝送速度の実験を行っている．これら 3 つのケースの実験結果より，NORM+ATRaCT の場合，どのケースよりも伝送速度が速く，受信機の台数が 5 台以上になると TCP シリアル送信が TCP 並列送信よりも早いという結果が報告されている [8].

3. 提案

本稿では，時間的効率の観点から UDP マルチキャストを用いたファイル送信を提案する．マルチキャストとは，送信者から複数の受信者へ同じデータを同時に配信する通信プロトコルである [9]．データ受信を希望するノードは，特定のマルチキャストグループに参加する必要がある．マルチキャストを利用することで，送信側は，一台分の送信負荷で複数の IoT デバイスに対するファイル送信が可能になる．ここでの送信側の送信負荷の差が送信完了時間の差に繋がる．まず，どのようにパケットロスしたパケットを特定するかの課題について，シーケンス番号を各パケットの先頭に付加する方法を提案する．

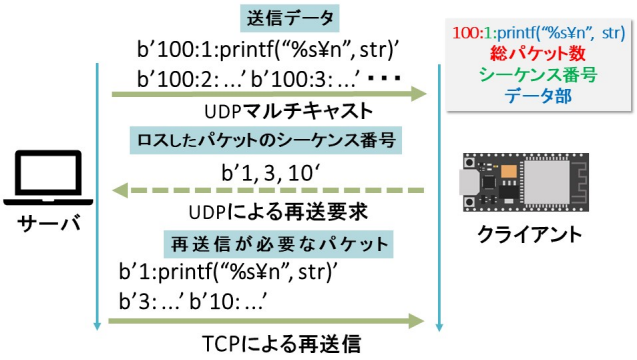

図 2　サーバとクライアントとの通信

図 2 が本稿で提案する UDP マルチキャストと TCP 並列送信を用いた信頼性のあるデータ通信の概要である．まず送信側 (以下: サーバ) はパケット送信時に，各パケットのデータ部の先頭に総パケット数とシーケンス番号とコロンを付加する．形体としては，' 総パケット数: シーケンス番号: データ部' の様になる．そのパケットを受け取った受信側 (以下: クライアント) は，コロンをデリミタとして分

解し，データ部とシーケンス番号を配列にそれぞれ格納していく．総パケット数は，全てのパケットが受信できたかの可否を判断するための条件に用いる．

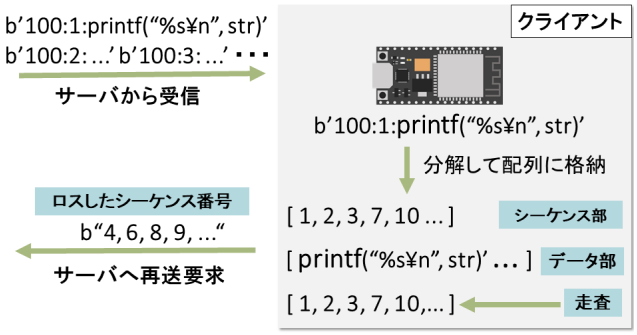

図 3　クライアント側の処理の細部

図 3 がクライアント側の処理の詳細である．クライアントは，シーケンス番号を格納した配列を昇順にソートし，順に走査していく．欠けた番号があった場合，それを飛び番とし，飛び番を記録した文字列を作成し，それをサーバに返送することで再送要求とする．クライアントからの再送要求を受け取ったサーバは，受け取った番号を基に，パケットロスをしたパケットを再生成し，クライアントに送り返す．この際，確実性を高めるために再送要求と再送信は TCP を用いて行う．また本提案のボトルネックは，パケットロスが起こった場合，これ以上新しいパケットが送られてこないことをクライアントが判断するためのタイムアウト時間を設定する必要があることである．この設定するタイムアウト時間によって TCP を用いた方法の送信完了時間との時間差が左右されるため，考慮が必要となる．これに関しては，タイムアウト時間を変更しながら実験し決定していく．

ユースケース・シナリオ

ユースケースとして図 4 のように，小・中規模の IoT を用いるプロジェクトに際する，IoT デバイスのファームウェアアップデートに本提案を用いることができる．

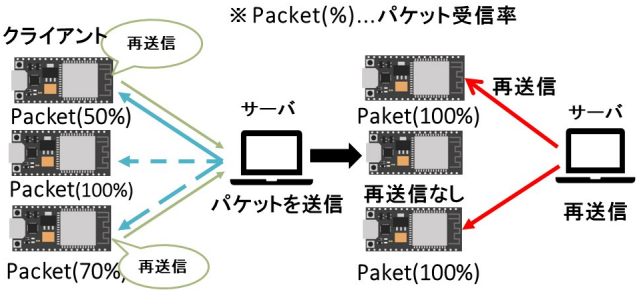

図 4　ファームウェアのアップデート

大規模なプロジェクトでは，LoRaWAN や Sigfox, NB-IoT といったプロトコルに代表される LPWA が主に運用されている．現在，現場に配置される IoT デバイスのライフサイクルは 10 年以上続くことが予想されるため，セキュリティの観点からファームウェアの定期的な更新は必須である [7]．セキュリティを考慮したファームウェアの更新では，複数の機器に対して，迅速にファイル送信を完了する必要があることに加え，パケットロスが生じてしまうことを妥協することはできない．そのため効率的かつ信頼性を保障した本提案は適当である．

4.　実装

サーバの OS として Ubuntu20.04，ソフトウェアは Python で実装する．クライアントである IoT デバイスは．Arduino IDE の開発環境を用いて，Arduino 言語 (.ino) で実装を行った．本稿での IoT デバイスとして，Espressif Systems 社の ESP32 を使用した．主に，プログラムは，”マルチキャスト送受信プログラム”と”再送信要求プログラム”，”再送プログラム”の 3 つに大別される．図 5 にて，実装図を示す．

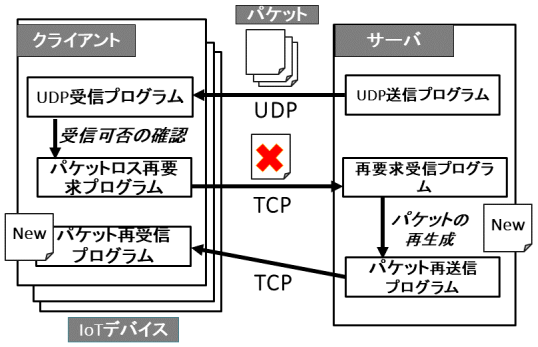

図 5　実装図

マルチキャスト送受信プログラム

サーバとクライアントは同じマルチキャストグループに属し，サーバはクライアントに対し，マルチキャストにてパケットの送信を行う．この際，パケットのデータ部に総パケット数，シーケンス番号を付加し送信する．

再送信要求プログラム

サーバから複数パケットを受け取ったクライアントは，シーケンス番号を別途，配列に格納する．その後，その配列を走査することで，パケットロスを把握し，サーバに再送要求を行う．ここでこれ以上パケットを受け取れないと判断するためのタイムアウト時間として 1 秒を設定した．

再送プログラム

サーバは，パケットロスしたシーケンス番号を確認し，

再送信が必要なパケットを再生成した後，TCP にて再送を行う．ここでサーバは，複数クライアントから一度に再送要求を受け取ることになるため，複数処理を同時に捌く必要がある．そのため，Python3.8.10 の threading モジュールを用いてマルチスレッドによる TCP 並列送受信を実装した．

5. 実験と分析

実験環境

図 6 が実験環境である．1 台のサーバから 1〜5 台までのクライアントに対し，ファイル送信を行い，再送要求に応じて再送信を行う．

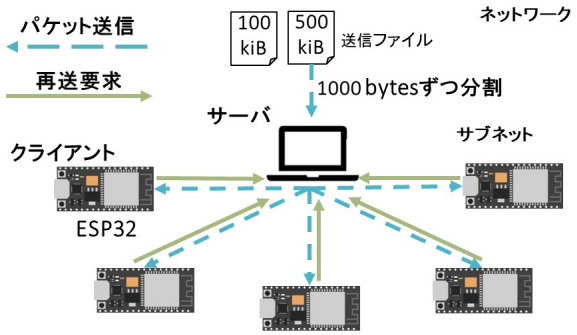

図 6 実験環境

評価方法

評価の指標として，TCP シリアル送信で行う複数 IoT デバイスへのファイル送信 (以下: TCP 送信) と本稿の提案である UDP マルチキャストと TCP 並列送信を組み合わせた方法 (以下: マルチキャスト送信) との送信完了時間をそれぞれ比較する．それぞれのケースにおいて，50 回ずつ計測行う，TCP 送信に関しては，マルチスレッドを用いた複数台への並列データ送信も考えられるが，シリアルに送信を行う方が並列送信での送信完了時間より早いという研究結果があるためここではシリアル送信を採用する [8]．評価の条件として，IoT デバイスの台数，ファイルサイズの観点から評価していく事とする．ファイルサイズはファームウェアを想定し，100KiB，500KiB としてそれぞれ実験を行う．

実験フローチャート

TCP 送信では，TCP ソケットの生成から最後の ESP32 に対しての送信が完了した時点までを送信完了時間とし，マルチキャスト送信では，複数 IoT デバイスにマルチキャストでファイル送信を行った時点から最後に再送要求した IoT デバイスへの再送が完了した時点までとする．各方法のフローチャートを以下図 7，8 に示す．ここでは，送信完了時間の把握のため，Python の time モジュールを使用し，送信開始時間から終了時間までを計測した．

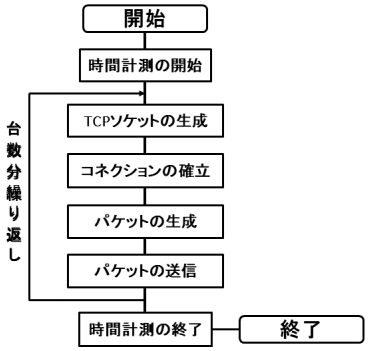

図 7 TCP 送信フローチャート

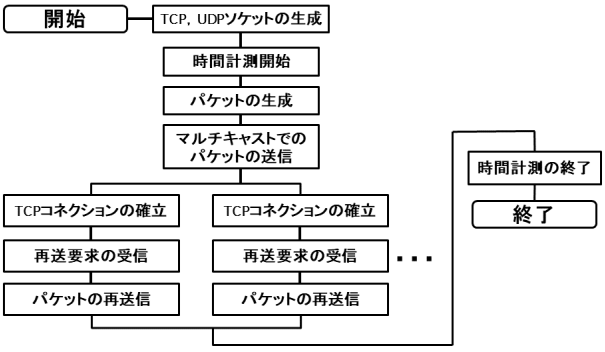

図 8 マルチキャスト送信フローチャート

図 8 に関して，TCP コネクションの確立は通信するデバイスの数だけ行うものである．

実験結果

・100KiB のファイル送信

図 9 は，TCP 送信でのデータ送信の結果である．

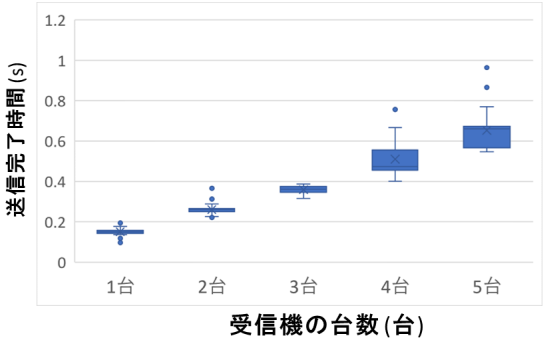

図 9 100KiB ファイルでの TCP 送信

平均送信完了時間は，各台数において，1 台: 0.15 秒，2 台: 0.26 秒，3 台: 0.36 秒，4 台: 0.51 秒，5 台: 0.65 秒という結果になった．ユニキャストで順次的に送信行っているため，1〜5 台の間において，に平均 0.126 秒ずつ線形増

加してくということが分かった．本稿では 5 台のみの計測であるが，Geithner, T らの研究より受信機が 5 台以上から伝送時間の増加は緩やかになっていくため，このまま伝送時間が線形増加するとは限らない [8]．だが台数が増加していく毎に送信時間は増加していくということが予想される．

図 10 は，マルチキャスト送信でのデータ送信の結果である．

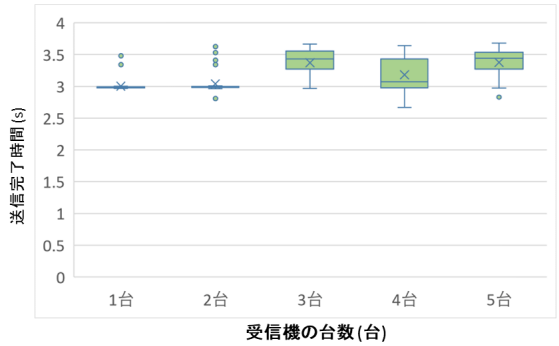

図 10　100KiB ファイルでのマルチキャスト送信

この時，これ以上パケットは到着しないと判断するタイムアウト時間は基準として 1 秒に設定した．グラフより，1 台〜2 台時の送信完了時間がやや早いが，どの台数においても 3〜3.5 秒の間に中央値が位置している．今回の条件での各台数毎の平均パケットロス率は表 1 の様になった．

表 1　100KiB でのパケットロス率

台数	パケットロス率
1 台	4.2%
2 台	5.4%
3 台	3.6%
4 台	2.8%
5 台	6.8%

ここでのパケットロスの平均値は，各機器ごとのパケットロス平均値を更に機器の台数分で平均を算出した値である．計算方法を具体的に図 11 に示す．

	1台目	2台目	3台目	4台目	5台目	
計測1回目	5	3	10	4	3	
2回目	3	1	4	1	2	
3回目	10	3	2	3	1	
⋮						
各機器の平均	6	2.3	5.3	2.3	2	計 17.9個

パケットロス数の総和	台数	平均パケットロス数	総パケット数	平均パケットロス率
17.9個	5台	3.58個	100個	約3.6%

図 11　パケットロス率の算出方法

タイムアウト時間を 0.5 秒に設定した際のパケットロス率は平均 28.7%という結果となった．0.5 秒の設定時間では，受け取れるはずのパケットが受け取れず，連番で多くのパケットが失われてしまった．そのためタイムアウト時間は，1 秒程度にする必要がある．

・500KiB のファイル送信

図 12 が，500KiB ファイルを扱った TCP 送信での結果である．1 台増えるごとに，平均 1.02 秒ずつ増加している．

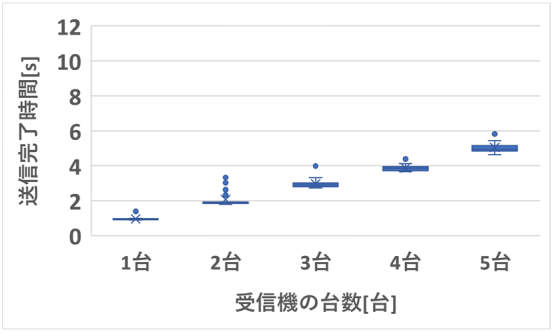

図 12　500KiB ファイルでの TCP 送信

図 13 が，500KiB ファイルを扱ったマルチキャスト送信での結果である．台数が増えても，10 秒程度でほぼ一定で

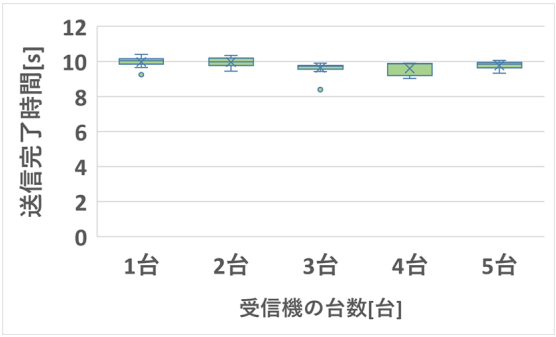

図 13　500KiB ファイルでのマルチキャスト送信

ある．今回の条件での各台数毎の平均パケットロス率は表 2 の様になった．

表 2　500KiB でのパケットロス率

台数	パケットロス率
1 台	12.2%
2 台	4.6%
3 台	8.5%
4 台	2.3%
5 台	3.6%

6. 議論

本提案では5台のみの実験を行った．その結果，UDPマルチキャスト送信はTCP送信と比べ，ファイルサイズが100KiBの場合，2.85秒 (受信機5台時点)，500KiBの場合，4.72秒 (5台時点) 遅くなってしまうことが分かった．UDPで送信する以上，1つ以上パケットロスする可能性が高いため，1秒間パケットを待ち続けてしまう．この1秒は本提案で設けたタイムアウト時間である．そのため差分時間の内1秒はタイムアウト時間，それ以外はUDPからTCPへ遷移する時間やパケットを生成する際に生じた時間である．

しかしながら，本提案では受信台数が増えたとしても送信完了時間が100KiBの場合，おおよそ3〜3.5秒の間に収まり，500KiBの場合，10秒程度に収まることが確認されたため，受信機の台数が増加していくほど，本提案の効果は増していく．100KiBの場合，TCP送信での送信完了時間が台数に応じて0.126秒ずつ線形増加していくと仮定した時，受信機が28台以上で本提案の方が早いということになる．500KiBの場合では，1.02秒ずつ増えていくと仮定した時，10台以上となる．これらの結果から受信機の台数とファイルサイズが増加するにつれ，本提案の方が早く送信を行えるということが分かる．

また本稿では評価基準として設けなかったが，マルチキャストによるトラフィックの改善も期待出来る．今後は，パケット損失率に応じた送信完了時間の変化やトラフィック量の比較，伝送距離といった点でも評価を行っていくことができる．

7. おわりに

本稿での課題として，複数のIoTデバイスへのファイル送信に関して，シリアルケーブル及びTCP，HTTPといったファイル送信では送信時間が多くなってしまう可能性があるという点を挙げた．それに対し，UDPマルチキャストとTCP並列送信を組み合わせたファイル送信を提案した．最初にマルチキャストにより，同じマルチキャストグループに属するIoTデバイスへ同時にファイル送信を行う．そしてパケットロスにより脱落したパケットを送信者に通知し，TCP並列送信を用いて再送信を行った．評価として，TCPシリアル送信のみを用いた場合のファイル送信と本稿でのUDPマルチキャストとTCP並列送信を組み合わせた方法との送信完了時間の比較を行った．その際の条件として1〜5台までの台数と100KiB, 500KiBのサイズで評価を行った．

謝辞

本稿へのアドバイスをしていただいた廣居舜佑さんには，感謝申し上げます．

参考文献

[1] Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions, *Future generation computer systems*, Vol. 29, No. 7, pp. 1645–1660 (2013).

[2] Shaukat, K., Alam, T. M., Hameed, I. A., Khan, W. A., Abbas, N. and Luo, S.: A review on security challenges in internet of things (IoT), *2021 26th International Conference on Automation and Computing (ICAC)*, IEEE, pp. 1–6 (2021).

[3] Aboubakar, M., Kellil, M. and Roux, P.: A review of IoT network management: Current status and perspectives, *Journal of King Saud University-Computer and Information Sciences* (2021).

[4] Haney, W. V.: Serial communication of information in organizations, *ETC: A Review of General Semantics*, pp. 13–29 (1964).

[5] Li, D., Xu, M., Liu, Y., Xie, X., Cui, Y., Wang, J. and Chen, G.: Reliable multicast in data center networks, *IEEE Transactions on Computers*, Vol. 63, No. 8, pp. 2011–2024 (2013).

[6] Cho, W.: LoRa for LPWA network: overview and its performance enhancement technologies, *The Journal of the Korea institute of electronic communication sciences*, Vol. 14, No. 2, pp. 283–288 (2019).

[7] Mahfoudhi, F., Sultania, A. K. and Famaey, J.: Over-the-Air Firmware Updates for Constrained NB-IoT Devices, *Sensors*, Vol. 22, No. 19, p. 7572 (2022).

[8] Geithner, T. and Sivrikaya, F.: Adaptive reliable multicast in 802.11 networks, *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, pp. 1–6 (2018).

[9] 長嶺進也ほか：オーバーレイネットワークにおけるマルチキャストプロトコルの研究，大学院研究年報 理工学研究科編， Vol. 41 (2011).