



دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

بهبود مصرف انرژی در سیستم‌های مبتنی بر اینترنت اشیا

گزارش سمینار کارشناسی ارشد مهندسی کامپیوتر
گرایش نرم افزار

نگارش

پویا یارندی

استاد راهنما

دکتر محمود نشاطی

تابستان ۱۳۹۹

چکیده

امروزه با افزایش قابلیت‌های مبتنی بر اینترنت، نیازمندی بیشتری برای اتصال اشیای پیرامون ما به شبکه اینترنت احساس می‌شود. از این رو، می‌توان گفت مفهوم اینترنت اشیا (Internet of Things) روز به روز در حال افزایش است و علاوه بر این، همواره به نیازمندی‌های مبتنی بر اینترنت اشیا افزوده می‌شود. بدیهی است که این افزایش نیازمندی‌ها به پیچیدگی بیشتر سیستم‌های مبتنی بر این فناوری می‌انجامد. بدین سبب می‌توان گفت رفته رفته با سیستم‌هایی مواجه می‌شویم که پیچیدگی بیشتری دارند. از مصادیق این پیچیدگی می‌توان به گستردگی و دسترس‌پذیری سیستم اشاره کرد. برای مثال؛ زمانی که ما به دنبال ایجاد یک سیستم مبتنی بر اینترنت اشیا برای یک شهر هوشمند هستیم، هزاران حسگر در این سیستم مشغول به مشاهده محیط خواهند بود و همچنین برای اینکه سیستم ارائه شده، سیستمی مورد اعتماد باشد، لازم است تا دسترس‌پذیری قابل قبولی داشته باشد. یکی از مهم‌ترین عوامل در دست یافتن به دو مورد بالا، بهبود مصرف انرژی در سیستم‌های مبتنی بر اینترنت اشیاست که می‌تواند عمر طولانی‌تر و همچنین هزینه نگهداری کمتری را برای سیستم به ارمغان آورد. در این تحقیق سعی داریم تا با بررسی کارهای صورت گرفته در این زمینه تکنیک‌های به کاررفته در سیستم‌های مبتنی بر اینترنت اشیا را شناسایی کنیم که از این موارد می‌توان به زمان‌بندی فعالیت حسگرها، ارائه توپولوژی بهینه برای شبکه، بهبود ارتباط حسگرها و سایر موارد اشاره کرد که در نهایت با مقایسه‌ای بین موارد بررسی شده عملکرد تکنیک‌های گوناگون را مورد ارزیابی قرار می‌دهیم.

واژگان کلیدی: اینترنت اشیا، بهینه‌سازی، مصرف انرژی

فهرست مطالب

۱	مقدمه	۱
۷	مروری بر ادبیات	۲
۸	DRX/DTX	۱.۲
۸	drxStartOffset	۱.۱.۲
۹	on-duration	۲.۱.۲
۹	drx-InactivityTimer	۳.۱.۲
۹	shortDRX-Cycle	۴.۱.۲
۱۰	longDRX-Cycle	۵.۱.۲
۱۰	drxShortCycleTimer	۶.۱.۲
۱۰	LEACH	۲.۲
۱۲	انواع حسگرها	۳.۲
۱۳	حسگرهای Trigger-Based	۱.۳.۲
۱۳	حسگرهای Periodic	۲.۳.۲
۱۴	خودسازماندهی	۴.۲
۱۵	خودپیکربندی	۱.۴.۲
۱۶	خودبهینگی	۲.۴.۲

۱۶	خودالتیامی	۳.۴.۲
۱۶	پارامترهای اندازه‌گیری	۵.۲
۱۶	Packet Loss Rate	۱.۵.۲
۱۷	Jitter	۲.۵.۲
۱۷	Average Sleep Ratio	۳.۵.۲
۱۷	IEEE 802.15.4	۶.۲
۱۸	IPv۶	۷.۲

۳ کارهای مرتبط ۲۰

۲۱	راهکارهای مبتنی بر مکانیزم DRX/DTX	۱.۳
۲۱	Counter Driven DRX (CDD)	۱.۱.۳
۲۴	نیرومندی راهکار ارائه شده	
۲۶	Multiple Threshold DRX (MTD)	۲.۱.۳
۲۸	DRX/DTX Optimized	۳.۱.۳
۲۸	تعیین زمان دوره	
۲۹	Inactivity Timer و On Duration مدت زمان	
۳۱	زمان بندی بسته‌ها	
۳۱	راهکارهای مبتنی بر گراف	۲.۳
۳۱	درخت EGF	۱.۲.۳
۳۲	مدل انرژی	
۳۳	ساخت درخت EFG	
۳۸	تجمیع کوئری‌ها	
۴۰	درخت ECH	۲.۲.۳
۴۱	راهکار (OGM) Object Group Mobility	۳.۳

۴۵ (SoT) Self Organized Things راهکار ۴.۳

۴۶ تابع ۱.۴.۳

۴۷ محاسبه هم‌پوشانی (ζ_i)

۴۸ محاسبه همبستگی فضایی (I_k)

۴۹ مراجع

فهرست شکل‌ها

۱.۲	شمای فعالیت مکانیزم DRX/DTX	۸
۲.۲	انتخاب گره‌ی سرگروه و خوشه‌بندی در دو بازه‌ی زمانی متفاوت توسط الگوریتم LEACH	۱۱
۳.۲	چرخه MAPE	۱۵
۱.۳	درخت موضوعی تحقیق	۲۲
۲.۳	افزایش دوره DRX در واکنش به کمبود درخواست برای گره	۲۳
۳.۳	کاهش دوره DRX در واکنش به ازدیاد درخواست برای گره	۲۴
۴.۳	خطاهای ممکن در حین برقرار ارتباط میان دو گره	۲۵
۵.۳	حالت‌های کیفیت در MTD و حد حساب هر یک	۲۷
۶.۳	محیط مورد بررسی در مقاله‌ی []	۳۳
۷.۳	شبه کد الگوریتم ساخت درخت EGF	۳۵
۸.۳	شبه کد الگوریتم محاسبه‌ی وزن میان سلول‌ها	۳۶
۹.۳	شبه کد الگوریتم محاسبه‌ی وزن میان زیرمحیط‌ها	۳۶
۱۰.۳	گراف حاصل شده برای محیط شکل ۶.۳	۳۷
۱۱.۳	محاسبه‌ی وزن میان دو سلول مجاور	۳۸
۱۲.۳	بازگرداندن داده به Base Station	۴۰
۱۳.۳	شبه کد الگوریتم ساخت درخت ECH	۴۲
۱۴.۳	شبه کد الگوریتم خوشه‌بندی ECH	۴۲

۴۳	۱۵.۳	گراف ساخته شده با الگوریتم ECH
۴۶	۱۶.۳	فرآیند جابجایی حسگرها در OGM
۴۷	۱۷.۳	شبه کد تابع Self-Configuration

فصل ۱

مقدمه

اینترنت اشیا^۱ یکی از فناوری‌های مورد توجه این روزها در دنیای کامپیوتر است. یکی از عواملی که سبب احساس نیازمندی به این فناوری نوپا گشته، اهمیت یافتن استفاده از اینترنت در دهه اخیر می‌باشد. امروزه نیازمندی انسان به اتصال و بهره‌مندی از شبکه جهانی اینترنت بر کسی پوشیده نیست و حتی بسیاری از نیازهای ابتدایی انسان از طریق اینترنت مرتفع می‌گردد. در چنین شرایطی، برای استفاده از انواع ماشین‌هایی که پیش از این نقش به‌سزایی در بهبود سطح زندگی انسان ایفا می‌کرده‌اند؛ نیازمندی جدیدی حس می‌شد که آن، فعالیت این ماشین‌ها بر بستر اینترنت بود. این امر باعث فعالیت یکپارچه‌تر دستگاه‌هایی می‌شد که پیش از این به طور مستقل از یکدیگر نیازمندی‌های بشر را تامین می‌کردند و بدین سبب، مفهوم اینترنت اشیا معرفی گردید.

با گسترش روزافزون نیازمندی‌های مبتنی بر اینترنت اشیا، نیازمندی‌ها و همچنین کاربردهای جدیدی برای این فناوری کشف می‌گردد. برای مثال امروزه شاهد استفاده از سیستم‌های مبتنی بر اینترنت اشیا در کاربردهایی گوناگون همچون ساختمان‌های هوشمند، شهرهای هوشمند و مواردی از این قبیل هستیم. به عنوان مثال، در یک سیستم مبتنی بر اینترنت اشیا که قرار است مدیریت یک شهر هوشمند را برعهده داشته باشد، هزاران حسگر^۲ در محیطی گسترده (در مقیاس کیلومتر مربع) مشغول به فعالیتند. در چنین شرایطی دو عامل اصلی به عنوان عوامل مورد بررسی جهت سنجش قابل اعتماد بودن^۳ سیستم مطرح است. این دو عامل دقت مشاهده^۴ محیط و همچنین میزان انرژی مصرفی محیط می‌باشند. در ادامه به بررسی این دو عامل و رابطه‌شان نسبت به یکدیگر می‌پردازیم.

یکی از عوامل ذکر شده در بالای برای بررسی میزان قابل اعتماد بودن سیستم‌های مبتنی بر اینترنت اشیا، دقت مشاهده محیط است. فرض کنید در یک خانه هوشمند، قرار است سیستم مراقبت از سالمندان در آن پیاده‌سازی شود. این سیستم باید به طور مداوم به مشاهده محیط پرداخته و در صورت بروز هرگونه رفتار غیرطبیعی در سالمند (از جمله افتادن بر روی زمین و...) اورژانس را از وضعیت به وقوع پیوسته مطلع سازد. علاوه بر این اطلاعاتی از قبیل فشار خون، تعداد ضربان و... نیز از شخص سالمند که در طول چند ساعت اخیر ضبط شده است به سیستم درمانی ارائه دهد. مشخص است که در چنین شرایطی و در چنین سیستمی اهمیت عامل اول،

^۱ Internet of Things

^۲ Sensor

^۳ Reliability

^۴ Observation

یعنی دقت سیستم بسیار زیاد است.

برای دستیابی به چنین هدفی لازم است تا حسگرها به طور مداوم و با فواصل زمانی کم به مشاهده محیط پرداخته و اطلاعات موردنیاز را از محیط جمع‌آوری نمایند. بدین ترتیب، اطلاعات به‌دست‌آمده از محیط، اطلاعات به‌روزتری خواهد بود و به همین دلیل دقت سیستم افزایش می‌یابد. البته به غیر از تناوب مشاهده محیط توسط حسگرها، عوامل دیگری همچون کیفیت خود حسگرها، نحوه نصب آن‌ها در محیط و سایر موارد نیز می‌تواند تاثیر به‌سزایی در دقت اندازه‌گیری مقادیر در محیط و در نتیجه مشاهده بهتر آن داشته باشد که البته این موارد در حیطه بحث این پژوهش نخواهد بود.

اما عامل دیگر در بررسی سیستم‌های مبتنی بر اینترنت اشیا که اتفاقاً مبحث اصلی این پژوهش نیز بوده، مصرف انرژی^۱ است. یک سیستم مبتنی بر اینترنت اشیا، شامل چندین حسگر است که در محیط مستقر بوده و به تناوب در حال دریافت اطلاعات محیط می‌باشند. بدیهی‌ست که دو موضوع در بحث تامین انرژی حسگرهای چنین سیستمی مطرح است. اول اینکه اگر حسگرهای این سیستم به‌طور بی‌رویه به مشاهده محیط و دریافت اطلاعات از آن پردازند، انرژی بسیار زیادی مصرف می‌شود که خود این امر تا حدود بسیاری نامطلوب است. به‌هرحال در دنیای امروز یکی از مباحث بسیار مهم و حائز اهمیت، بحث مصرف بی‌رویه منابع انرژی است و واضح است که در صورت بهینه نبودن سیستم‌های مبتنی بر اینترنت اشیا به لحاظ مصرف منابع انرژی، این سیستم‌ها نخواهند توانست جای خود را در زندگی روزمره ما بیابند.

علاوه بر این، موضوع دیگری که اهمیت مصرف بهینه منابع انرژی را برای سیستم‌های مبتنی بر اینترنت اشیا دو چندان می‌کند، بحث طول عمر این سیستم‌ها است. با توجه به ماهیت سیستم‌های مبتنی بر اینترنت اشیا، معمولاً این سیستم‌ها به لحاظ گستره جغرافیایی سیستم‌هایی بسیار بزرگ هستند که گاهی مساحت تحت پوشش آن‌ها می‌تواند به بزرگی یک شهر باشد (همانند مثال شهر هوشمند). بنابراین آنچه حائز اهمیت است، این است که امکان تامین انرژی تمام سیستم به صورت متمرکز ممکن است وجود نداشته باشد. از این روی، رویکرد اصلی در این سیستم‌ها این است که حسگرها خود توانایی تامین انرژی خود را به طور مستقل داشته باشند. یکی از راهکارها در این زمینه، استفاده از باتری برای تامین انرژی حسگرهاست که در جهت مستقل بودن حسگرها

^۱Energy consumption

و همچنین افزایش قابلیت جابه‌جایی^۱ آن‌ها صورت می‌گیرد. بنابراین منابع انرژی که در اختیار هر حسگر قرار می‌گیرد، محدود است و استفاده بهینه از این منابع بسیار اهمیت می‌یابد. هر چند این منابع قابل تمدید هستند (یعنی می‌توان باتری یک حسگر را با باتری جدید معاوضه نمود.) اما آنچه حائز اهمیت است، در واقع فعال نگه داشتن سیستم برای مدت طولانی‌تری است. به طور کلی می‌توان طول عمر سیستم را یکی از مزایای بهینه‌سازی مصرف منابع انرژی دانست که منجر به قابل اعتماد بودن بیشتر سیستم می‌گردد.

بنابراین همان‌گونه که مشخص است یک Trade-Off در بین دقت عملکرد سیستم و طول عمر سیستم وجود دارد. به این معنی که اگر بخواهیم دقت سیستم را افزایش دهیم، حسگرها باید با تناوب‌های زمانی کوتاه‌تری به مشاهده و دریافت اطلاعات محیط پردازند و در نتیجه انرژی بیشتری مصرف کنند که این امر منجر به کاهش طول عمر حسگرها و در نهایت کاهش طول عمر سیستم می‌گردد. و در طرف مقابل اگر در صدد افزایش طول عمر سیستم‌های مبتنی بر اینترنت اشیا باشیم، بدیهی است که باید تناوب زمانی فعالیت حسگرها طولانی‌تر گردد. این امر باعث می‌شود تا دریافت اطلاعات از محیط با فواصل زمانی بیشتری از جانب حسگرها صورت پذیرد و در نتیجه دقت اطلاعات حاصله از محیط کاهش یابد. چرا که ممکن است تغییراتی در محیط، در فواصل زمانی‌ای رخ دهد که حسگرها مشغول به فعالیت نیستند و در نتیجه برخی اطلاعات در ارتباط با رویدادهای رخ داده در محیط از دست بروند.

همان‌طور که مشاهده شد، یک تضاد در میان دو عامل اصلی جهت به‌دست‌آوردن اعتماد برای استفاده از سیستم‌های مبتنی بر اینترنت اشیا وجود دارد. آن‌چه در این تحقیق به آن پرداخته شده است در واقع پدید آوردن توازنی نسبی در میان این دو عامل است. به طوری که یک سیستم مبتنی بر اینترنت اشیا هم دارای دقت قابل قبولی در مشاهده و دریافت اطلاعات مربوط به رویدادهای رخ داده در محیط باشد؛ و هم میزان مصرف منابع انرژی آن بهینه باشد و اتلاف منابع انرژی را به حداقل برساند. برای این منظور پژوهش‌هایی پیش از این صورت گرفته است که ما در این تحقیق به آن‌ها می‌پردازیم. اما به طور کلی چند رویکرد کلی در دستیابی به این امر وجود دارد که در ذیل به شرح مختصری از هر کدام اکتفا می‌کنیم و در بخش‌های بعدی به تفصیل آن‌ها را بررسی می‌نماییم.

لازم به ذکر است که ایده‌هایی همچون بهبود سخت‌افزاری و بهینه‌سازی‌هایی از این قبیل نیز در پژوهش‌های

^۱Portability

پیش از این مطرح شده است و اما در این تحقیق، تمرکز اصلی ما بر روی راهکاری نرم افزاری و معمارانه معطوف شده است. بنابراین از تشریح راهکارهای سخت افزاری صرف نظر می نماییم.

یکی از رویکردهای پرتکرار در میان مقالات، استفاده از زمانبندی برای فعالیت حسگرهاست. به این معنی که زمانهای مورد نیاز برای فعالیت یک حسگر مشخص می شود و سپس حسگر در این زمانها شروع به فعالیت می کند. پس از آن که طبق برنامه زمانبندی شده، زمان فعالیت حسگر به پایان رسید حسگر غیرفعال شده و تا بازه‌ی زمانی بعدی که باید فعالیتش را شروع کند مصرف انرژی خود را به حداقل می رساند. این امر اغلب با غیرفعال شدن مدار ارتباطی حسگر صورت می گیرد؛ چراکه بخش زیادی از منابع انرژی مصرفی توسط یک حسگر، صرف برقراری ارتباط با دیگر اجزای سیستم می گردد. بدیهی است که یکی از دلایل این مصرف زیاد برای برقراری ارتباط این است که اغلب ارتباطات در سیستمهای مبتنی بر اینترنت اشیا به صورت بی سیم^۱ و با استفاده از تجهیزات ارتباطی نظیر Bluetooth صورت می پذیرد. در مقالات مختلف دستیابی به یک سیستم زمانبندی برای فعالیت حسگرها، به کمک ابزارهای گوناگون از جمله ارائه معماری برای سیستمهای مبتنی بر اینترنت اشیا و یا استفاده از مکانیزم های از پیش تعریف شده تحقق یافته است.

یکی دیگر از رویکردهای ارائه شده در مقالات مورد بررسی، بحث ارائه یک ساختار جهت برقراری ارتباط میان حسگرها و سایر اجزای سیستم است به طوری که ارتباط آنها با هزینه کمتری به لحاظ مصرف انرژی صورت گیرد. این امر به کمک پیدا کردن مسیرهای کوتاه تر در میان اجزای سیستم میسر می شود و در واقع اساس کار این رویکردها این است که ارتباط میان اجزای نزدیک به یکدیگر با صرف انرژی کمتری صورت می پذیرد. همچنین در این رویکرد، راهکاری برای تجمیع اطلاعاتی که قرار است مورد مخابره قرار گیرد در نظر گرفته شده که از مصرف انرژی بی رویه در جهت انتقال اطلاعات در بستر شبکه اینترنت اشیا جلوگیری می نماید.

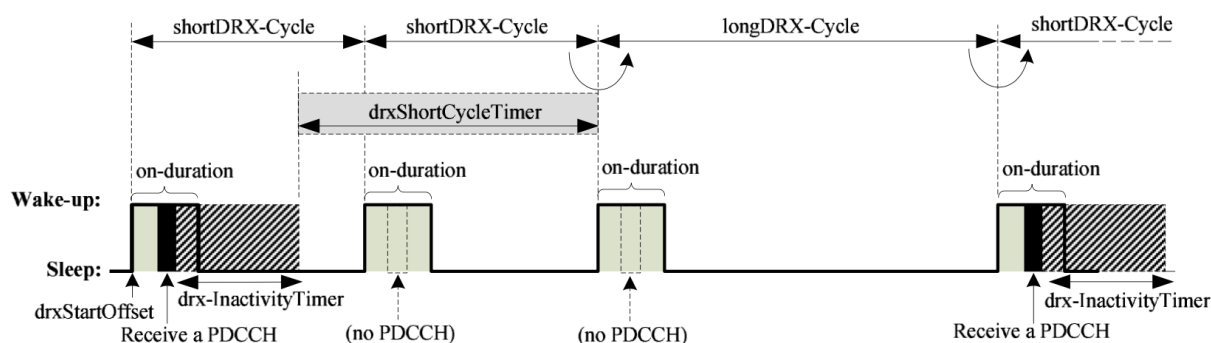
در مجموع بعد از بررسی راهکارهای ارائه شده در این پژوهشها مشاهده می شود که مصرف انرژی در سیستمهای مبتنی بر اینترنت اشیا تا مقدار قابل توجهی کاهش یافته و به طور مثال میزان انرژی مصرفی یک سیستم مبتنی بر اینترنت اشیا با ۴۵۰۰ حسگر پس از ۱۲۰ ساعت فعالیت به عددی در حدود $10 \times 10^6 nJ$ کاهش یابد که این مقدار حتی در قیاس با برخی روشهای دیگر بهینه سازی چیزی در حدود ۱/۴ انرژی مصرفی را مورد استفاده قرار می دهد.

^۱ Wireless

در بخش بعدی گزارش به مروری بر ادبیات تحقیق پرداخته و همچنین اصطلاحات مورد نیاز و همچنین پیش‌زمینه‌های لازم جهت درک بهتر راهکارهای ارائه شده را تشریح می‌نماییم. پس از آن در بخش سوم، به تفصیل، راهکارهای ارائه شده در مقالات مورد بررسی را تشریح کرده و با درخت موضوعی مبحث آشنا خواهیم شد تا ذهنیت مورد نیاز برای زمینه تحقیق را درک نماییم. پس از آن در بخش چهارم به نتیجه‌گیری پرداخته و راهکارهای ارائه شده را با هم مقایسه می‌نماییم و به بیان نقاط ضعف و قوت هریک می‌پردازیم.

فصل ۲

مروری بر ادبیات



شکل ۱.۲: شمای فعالیت مکانیزم DRX/DTX

در این بخش سعی داریم تا با مروری بر اصطلاحات و ابزارهای مورد استفاده در پژوهش‌های بررسی شده، با پیش‌نیازهای مبحث موردنظر آشنا شویم.

۱.۲ DRX/DTX

Discontinuous Reception/Transmission یا به اختصار DRX/DTX یک مکانیزم از پیش تعریف شده بر بستر شبکه‌های بی‌سیم است که به کمک آن می‌توان زمانبندی فعالیت هر یک از اعضای شبکه (که به اصطلاح، به آن گره^۱ نیز گفته می‌شود) ساماندهی نمود. همان‌گونه که در شکل ۱.۲ مشاهده می‌نمایید؛ در شبکه‌هایی که از این مکانیزم استفاده می‌شود در واقع یک چهارچوب خاص برای فعالیت گره‌ها وجود دارد و رفتار ثابت و از پیش تعریف شده‌ای را از گره‌ها شاهد هستیم. اما آن چه تفاوت در میان گره‌ها را رقم می‌زند مقادیر پارامترهایی است که در این مکانیزم به کار می‌روند. پارامترهای DRX/DTX هر یک، بیانگر زمان مورد نیاز برای یک رویداد در رفتار گره‌ها هستند. در ادامه به توضیح هر یک از این پارامترها می‌پردازیم.

۱.۱.۲ drxStartOffset

این پارامتر بیانگر Offset زمانی اولیه‌ی پیش از شروع فرآیند توسط گره‌ی شبکه می‌باشد. بدیهی است که تا پیش از این زمان، گره هنوز فعالیت خود را آغاز نکرده است.

^۱Node

۲.۱.۲ on-duration

این پارامتر نمایان گر مدت زمانی است که پس از هر بار فعال شدن، گره فعال می ماند و در انتظار دریافت داده از سایر گره ها می باشد. گفتنی ست که این مدت زمان فعال ماندن یک گره بر حسب نیاز می تواند افزایش یابد که در ادامه به آن خواهیم پرداخت.

۳.۱.۲ drx-InactivityTimer

پیش تر گفتیم که یک گره ی شبکه، در زمان خاصی از حالت غیرفعال خارج شده و در انتظار دریافت داده از سایر گره های موجود در شبکه می ماند. این زمان، در واقع همان زمانی است که در بخش پیش به تعریف آن پرداختیم. اما در آنجا نیز اشاره شد که در صورت دریافت سیگنال از سایر گره ها، نیازمند زمان فعالیت بیشتری برای گره هستیم. دلیل آن نیز واضح است. چرا که اگر در اواخر زمان on-duration، بسته ای (سیگنالی) توسط گره ی مذکور دریافت شود، طبیعتاً پردازش و در صورت نیاز، پاسخ دهی به آن نیازمند زمان است. پس طبق آنچه از مرور چنین شرایطی درک می نماییم؛ پس از دریافت یک سیگنال، گره نیازمند است تا زمان اضافه تری برای فعالیت در اختیار داشته باشد که این زمان اضافه را، drx-InactivityTimer می نامند. لازم به ذکر است که در این زمان، همان طور که از نام این پارامتر نیز مشهود است؛ گره امکان پاسخگویی به درخواست دیگری را ندارد.

۴.۱.۲ shortDRX-Cycle

تمام فرایندهای ذکر شده در بخش های بالا، در یک تناوب مشخصی تکرار می شوند که پارامتر shortDRX-Cycle بیان گر مدت زمان یک دوره می باشد. همان طور که در شکل ۱.۲ نیز مشاهده می شود، طول دوره، به از زمان پارامتر on-duration بیشتر خواهد بود. چرا که در هر دوره گره پس از شروع شدن زمان فعالیت، مدتی را منتظر دریافت سیگنال از جانب سایر گره ها می ماند. پس از این زمان، گره باید قادر باشد تا مدتی را غیرفعال گردد و میزان مصرف انرژی سیستم را کاهش دهد. از این رو، بدیهی است که مدت زمان دوره از زمان فعالیت دستگاه طولانی تر باشد.

۵.۱.۲ longDRX-Cycle

پیش از این با مفهوم دوره در شبکه‌های مبتنی بر مکانیزم DRX/DTX آشنا شدیم. اما یکی دیگر از جنبه‌های قابل توجه در این مکانیزم، وجود دوره با دو مدت زمان متفاوت است. در این مکانیزم، دو دوره با مدت زمان‌های کوتاه و بلند وجود دارد. فعالیت دستگاه در این سیستم‌ها با دوره کوتاه آغاز می‌شود اما در صورتی که مدتی دستگاه پس از فعال شدن و انتظار دریافت سیگنال، هیچ سیگنالی دریافت نکند؛ در این شرایط وارد دوره‌های بلند مدت می‌شود. از آن‌جا که تمامی پارامترهای ذکر شده در بالا برای هر دو طول دوره یکسان است، می‌توان نتیجه گرفت که دوره طولانی‌تر به مدت زمان غیرفعال بودن طولانی‌تری برای دستگاه می‌انجامد و به همین ترتیب میزان مصرف انرژی گره بیش از پیش کاهش می‌یابد.

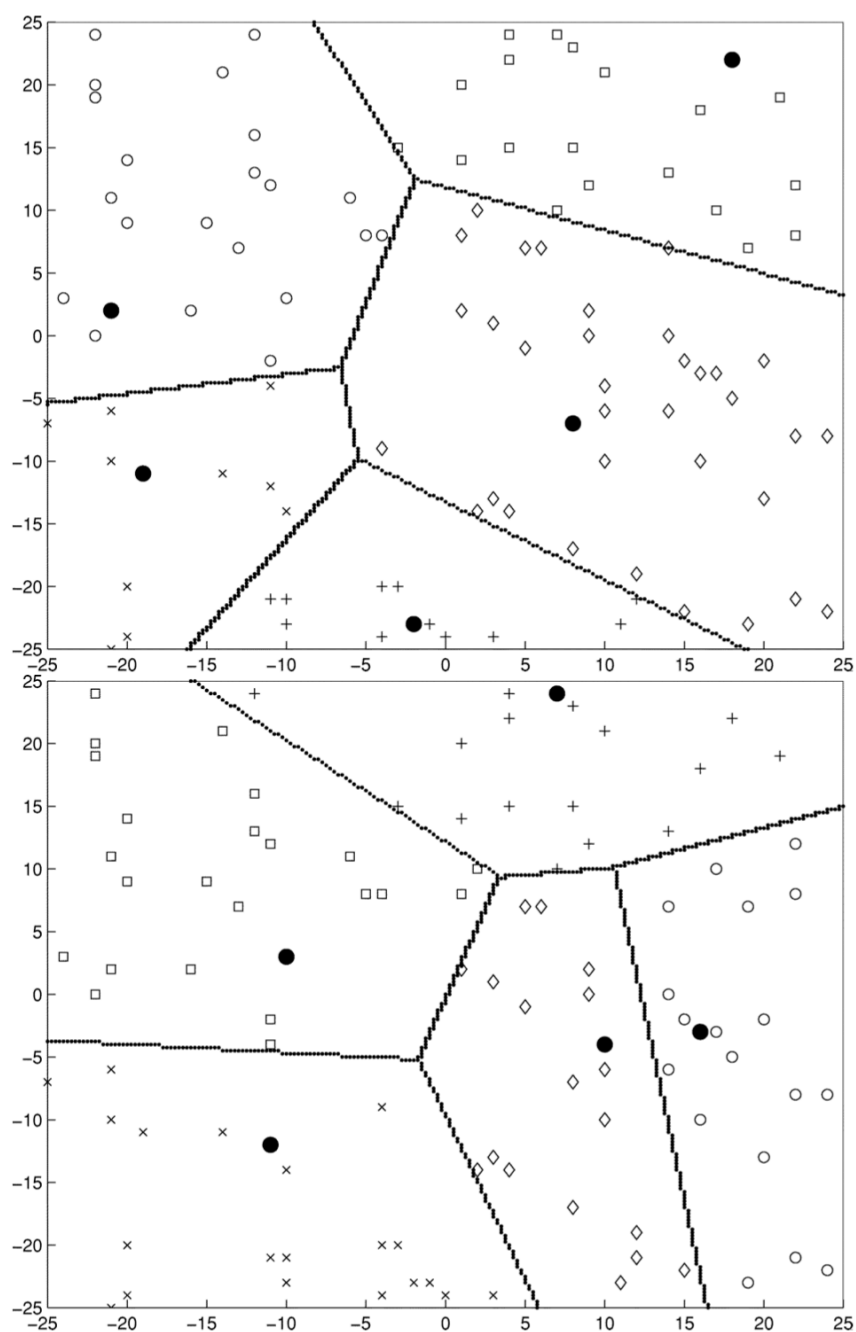
۶.۱.۲ drxShortCycleTimer

این پارامتر بیانگر مدت زمانی است که پس از آن، دستگاه از دوره‌های کوتاه مدت وارد دوره‌های بلند مدت می‌شود تا انرژی کمتری مصرف نماید.

۲.۲ LEACH

در مبحث کارهای صورت گرفته، در ادامه خواهیم دید که یکی از رویکردها جهت کاهش مصرف منابع انرژی توسط حسگرها در سیستم‌های مبتنی بر رایانش ابری، ارائه گراف یا توپولوژی‌ای برای مشخص نمودن حالت بهینه‌ی ارتباط میان اجزا می‌باشد. در این رویکردها یکی از تکنیک‌های به کار برده شده این است که به دلیل ساختار سلسله‌مراتبی‌شان، در هر حالت، گره‌ای به عنوان گره اصلی خوشه^۱ انتخاب می‌گردد. به طور خلاصه اگر به بیان وظیفه این گره نگاه کنیم؛ در واقع این گره مسئولیت انتقال داده از سطوح بالاتر همچون Base Station به حسگرها و همچنین جمع‌آوری داده‌ها از حسگرها و ارسال مجدد آن‌ها برای Base Station را بر عهده دارد. در چنین شرایطی بدیهی است که گره‌ای که به عنوان گره سرگروه خوشه انتخاب می‌گردد؛ دارای فعالیت ارتباطی بیشتری با سایر گره‌ها دارد و در نتیجه منابع انرژی بیشتری را نسبت به سایر گره‌ها مصرف می‌نماید.

^۱ Head node



شکل ۲.۲: انتخاب گرهی سرگروه و خوشه‌بندی در دو بازه‌ی زمانی متفاوت توسط الگوریتم LEACH

حال اگر فرض کنیم فرآیند انتخاب گره سرگروه یک فرآیند ثابت باشد - یعنی فقط یک بار گره‌ای را به عنوان سرگروه مشخص کنیم و دیگر آن را تغییر ندهیم - آن‌چه به وقوع می‌پیوندد، این است که در هر خوشه یا منطقه‌ای که مستقلاً به عنوان یک گروه در نظر گرفته شده است، یک گره (که همان گره سرگروه است) با مصرف حداکثری انرژی به کار خود ادامه می‌دهد تا زمانی که منابع انرژی خود را به طور کامل مصرف کرده و خاموش گردد. در این رویکرد همان‌طور که قابل مشاهده است، طول عمر سیستم کاهش می‌یابد و این دقیقاً در تناقض با یکی از اهداف اصلی ما، یعنی افزایش طول عمر سیستم است.

به همین منظور، الگوریتمی که در مقاله [] معرفی شده است، الگوریتم LEACH (مخفف‌شده‌ی عبارت: Low-Energy Adaptive Clustering Hierarchy) می‌باشد. رویکرد این الگوریتم به این صورت است که به صورت دوره‌ای، گره سرگروه هر خوشه را تغییر می‌دهد تا سربار انرژی حاصل از برقراری ارتباط میان اجزای سیستم، به طور متوازن میان تمامی گره‌های عضو خوشه تقسیم گردد و در نتیجه، به افزایش طول عمر سیستم منجر شود. در شکل ۲.۲ نمونه‌ای از رفتار این الگوریتم را در دو دوره‌ی متفاوت زمانی مشاهده می‌کنید.

۳.۲ انواع حسگرها

همان‌طور که می‌دانیم؛ حسگرهای به کار رفته در یک سیستم مبتنی بر اینترنت اشیا، عمدتاً مسئولیت مشاهده و دریافت اطلاعات از محیط را بر عهده دارند و این امر به واسطه‌ی ارتباطشان با Base Station و کوئری درخواست شده از جانب آن صورت می‌گیرد. بر این اساس، حسگرهای مبتنی بر اینترنت اشیا، از لحاظ رفتار و نحوه فعالیتشان به دو دسته اساسی تقسیم می‌شوند. این دو دسته عبارتند از:

- حسگرهای Trigger-Based

- حسگرهای Periodic

در ادامه به تشریح هر یک از انواع مذکور می‌پردازیم.

۱.۳.۲ حسگرهای Trigger-Based

دسته‌ی اول حسگرهای موجود در سیستم‌های مبتنی بر اینترنت اشیا، حسگرهایی هستند که بر اساس یک رویداد خاص در محیط فعال شده و به مشاهده محیط پیرامون می‌پردازند. برای مثال؛ حسگرهای حرکتی از این دست هستند و از کاربردهای مرتبط با آن، می‌توان به سیستم‌های روشنایی خودکار محیط اشاره کرد. در این سیستم‌ها با ایجاد حرکت در محیط، حسگر فعال می‌شود و سیگنالی را به Base Station تا کنترل‌کننده^۱ سیستم، دستور روشن شدن چراغ‌های موجود در محیط را ارسال کرده و روشنایی محیط تامین شود.

۲.۳.۲ حسگرهای Periodic

دسته دیگر حسگرهای موجود در سیستم‌های مبتنی بر اینترنت اشیا، حسگرهای Periodic هستند. این حسگرها، فعالیتشان منوط به یک رویداد خاص در محیط نیست و به طور مداوم در حال پایش و مشاهده محیط هستند. برای مثال حسگرهای دمای محیط اغلب به صورت Periodic فعالیت می‌کنند. چرا که تغییرات دما در یک محیط عمدتاً به صورت پیوسته بوده و همواره در جریان است. اما به این دلیل می‌گوییم اغلب، چرا که حسگرهای دما نیز می‌توانند به صورت Trigger-Based فعالیت کنند و برای مثال؛ صرفاً بر اساس تجاوز دما از یک درجه‌ی خاص و یا کاهش دما به میزانی کمتر از حد معین، سیگنال موردنظر را به Base Station مخابره نماید تا سیستم، تغییرات خود را بر محیط اعمال کند.

بدیهی است که در یک سیستم مبتنی بر اینترنت اشیا، هر دو نوع حسگرها می‌توانند در کنار یکدیگر به فعالیت بپردازند. یک مثال معروف در این زمینه که در مقاله [۱] آمده است، مثال مدیری است که برای جلسه فردا برنامه‌ریزی می‌کند. در این مثال بیان می‌شود که یک مدیر، برای جلسه فردای خود که در ساعت هشت صبح است ساعتش را کوک می‌کند تا در ساعت ۷ صبح بیدار شود. در این بین، در ساعت ۶:۳۰ صبح گوشی مدیر با چک کردن داده‌های مربوط به ترافیک متوجه ترافیک سنگین می‌شود و مدیر را زودتر بیدار می‌کند. پنج دقیقه بعد، زمانی که مدیر در حال استحمام است؛ سیستم، دستور فعال شدن را برای دستگاه چای‌ساز صادر می‌کند تا در هنگام آماده شدن مدیر چای، آماده باشد و چای‌ساز در زمانی که چای آماده می‌شود، سیگنالی را به سیستم می‌فرستد

^۱ Controller

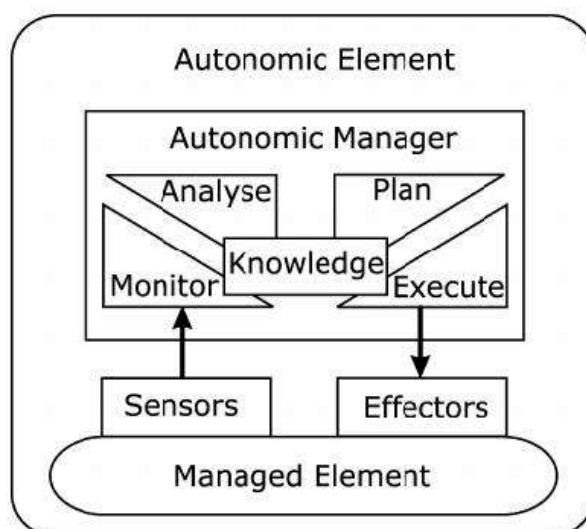
تا سیستم از این رویداد آگاه گردد. در این همین مثال مشاهده می‌شود که هر دو نوع حسگرهای Trigger-Based و Periodic در حال فعالیت در کنار یکدیگر می‌باشند. برای مثال؛ حسگر موجود در دستگاه چای‌ساز به صورت Trigger-Based عمل می‌نماید. حال آن‌که گوشی - که به عنوان حسگر ترافیک در سیستم مشغول به کار است - به صورت Periodic فعال است. نکته دیگر در تفاوت میان این دو نوع حسگر این است که به طور معمول، حسگرهای Periodic اطلاعات کاملی را از محیط مورد پایش خود به کنترل‌کننده ارائه می‌کنند. در صورتی که در اغلب موارد، حسگرهای Trigger-Based صرفاً یک سیگنال را که بیانگر وقوع رویداد است مخابره می‌کنند و این سیگنال شامل پارامترهای اضافی نیست.

۴.۲ خودسازماندهی

یکی از مفاهیم به کار رفته در مقاله [۱] که در فصل بعدی به تفصیل به آن خواهیم پرداخت، خودسازماندهی^۱ است. خودسازماندهی به معنی توانایی سیستم به ایجاد تغییر در رفتار خود به منظور بهبود عملکرد سیستم بوده و شامل جنبه‌های گوناگونی است که در ادامه به توضیح مختصری از هر یک می‌پردازیم. لازم به ذکر است؛ تعاریف ارائه شده در مقاله مذکور هیچ یک به تنهایی چرخه MAPE را به وجود نمی‌آورند. اما از مجموعه این اعمال چرخه MAPE پدید خواهد آمد. چرخه MAPE یک چرخه تکاملی است که در سیستم‌های خودسازمانده وجود دارد و حرکت سیستم در راستای این چرخه منجر به خودمختاری سیستم می‌گردد. این چرخه شامل چهار مرحله زیر است. در شکل ۳.۲ می‌توانید این چرخه را مشاهده نمایید.

- *Monitoring* به معنی مشاهده و پایش محیط و دریافت اطلاعات از آن.
- *Analyze* به معنی دریافت اطلاعات و پردازش آن‌ها که منجر به درک سیستم از محیط می‌گردد.
- *Planning* به معنی تصمیم‌گیری برای ایجاد تغییرات بر روی محیط براساس دانش موجود در ارتباط آن.
- *Execution* اعمال تغییرات برنامه ریزی شده در مرحله قبلی براساس ارسال دستورات به عمل‌کننده‌های محیطی.

^۱Self-Configuration



شکل ۳.۲: چرخه MAPE

در مقاله مذکور سه اصل جهت ایجاد چنین چرخه‌ای ذکر شده است که این مراحل به شرح زیر می‌باشند.

- خودپیکربندی^۱
- خودبهینه‌گی^۲
- خودالتیامی^۳

۱.۴.۲ خودپیکربندی

خودپیکربندی یک ویژگی در سیستم‌های خودمختار است که به واسطه‌ی آن، یک موجودیت جدید که در این سیستم وارد می‌شود نیازمند تنظیمات اولیه برای قرار گرفتن در جریان کاری سیستم نیست و توسط خود سیستم پیکربندی اولیه برای آن صورت می‌گیرد. این ویژگی به کاهش خطای انسانی به هنگام پیکربندی اولیه اجزا و البته کاهش پیچیدگی در سیستم منجر می‌شود.

^۱Self-Configuration

^۲Self-Optimization

^۳Self-Healing

۲.۴.۲ خودبهبینی

خودبهبینی، ویژگی دیگری در سیستم‌های خودمختار است که به واسطه‌ی آن، سیستم با اعمال تغییرات در پارامترهای موجود در اجزا، سعی در بهینه‌سازی عملکرد کلیت سیستم خواهد داشت. در مقاله [۱۱]، این شاخصه، به کمک یک تابع پیاده‌سازی شده است که وظیفه‌ی آن تعیین وضعیت فعالیت هر حسگر می‌باشد. در فصل بعدی به تفصیل این تابع را مورد تشریح قرار خواهیم داد.

۳.۴.۲ خودالتیامی

دیگر ویژگی سیستم‌های خودمختار، خودالتیامی می‌باشد. خودالتیامی بیان‌گر رفتاری در سیستم است که منجر به یافتن خطاهای سیستم و رفع کردن آن‌ها می‌گردد. در مقاله مورد بحث پس از طی مراحل قبلی که منجر به پیکربندی و سپس تعیین حالت حسگرها می‌شود؛ این ویژگی منجر به اصلاح وضعیت هریک از حسگرها در صورت لزوم خواهد شد. این تابع نیز در فصل بعد به طور کامل مورد بررسی قرار خواهد گرفت و با عملکرد آن آشنا خواهیم شد.

۵.۲ پارامترهای اندازه‌گیری

در سیستم‌های مبتنی بر اینترنت اشیا، و اصولاً به طور کلی، در هر سیستم پیاده‌سازی شد؛ یکی از مهم‌ترین دغدغه‌ها در ارتباط با صحت عملکرد سیستم پیشنهادی است. در این میان، در بین مقالاتی که مورد بررسی قرار گرفت، تعدادی از این پارامترها به عنوان پارامترهای اندازه‌گیری در جهت اثبات کارکرد راهکار پیشنهادی مورد استفاده قرار گرفته است. در ادامه این بخش به تشریح برخی از این پارامترها می‌پردازیم.

۱.۵.۲ Packet Loss Rate

یکی از پارامترهای اندازه‌گیری که در بررسی عملکرد سیستم‌های مبتنی بر اینترنت اشیا به کار می‌رود؛ Packet Loss Rate است. همان‌طور که می‌دانیم، اساس کار سیستم‌های مبتنی بر اینترنت اشیا، ارسال و دریافت پیام میان حسگرها و واحد کنترل‌کننده است. این پیام‌های در قالب بسته^۱هایی از داده بین اجزای سیستم منتقل

^۱Packet

می‌گردند و بنابر برخی عوامل ممکن است ارسال و یا دریافت آنان از جانب فرستنده و یا گیرنده دچار مشکل شود که در چنین حالتی ارسال پیام ناموفق بوده است. این پارامتر، بیانگر نسبت پیام‌های ارسالی ناموفق به مجموع پیام‌های مخابره شده در سیستم است.

۲.۵.۲ Jitter

یکی دیگر از پارامترهای به کار رفته در مقالات مورد بررسی، Jitter [۱] می‌باشد. Jitter در واقع، از محاسبه انحراف معیار مدت‌زمان دریافت بسته‌های ارسالی، به دست می‌آید. به عبارتی می‌توان دلیل اهمیت این پارامتر را در این دانست که اگر در بین مدت‌زمان‌های ثبت شده، بنابر هر دلیلی یکی از بسته‌ها با تاخیر قابل توجهی در دریافت مواجه شد؛ این امر کل عملکرد سیستم را از لحاظ ارزیابی تحت‌الشعاع قرار ندهد.

۳.۵.۲ Average Sleep Ratio

یکی دیگر از پارامترهای مورد استفاده، میانگین نسبت غیرفعال بودن^۱ سیستم است. همان‌گونه که از نام این پارامتر مشخص است؛ از میانگین نسبت زمان غیرفعال بودن حسگرها به نسبت به کل زمان فعالیت سیستم به دست می‌آید.

لازم به ذکر است که پارامترهای اندازه‌گیری نام‌برده شده در بالا، تمام پارامترهای مورد استفاده در مقالات مورد بحث نیستند. ما در این جا پارامترهایی را مورد بحث قرار داده ایم که نیازمند تشریح بیشتر بوده اند و در صورت نیاز، بقیه‌ی پارامترها را در فصل بعد، به طور ضمنی و به طور خلاصه تعریف می‌نماییم.

۶.۲ IEEE 802.15.4

IEEE 802.15.4 یک استاندارد ارتباطی کوتاه‌برد همانند Bluetooth است با این تفاوت که به لحاظ مصرف انرژی بهینه‌تر از این فناوری می‌باشد. این استاندارد که لایه‌ی فیزیکی و Media Access Control را در سیستم مشخص می‌نماید؛ به طور ذاتی برای کار با پروتکل آدرس‌دهی IPv4 طراحی شده است. اما برای این که قابلیت استفاده از پروتکل آدرس‌دهی IPv6 را نیز داشته باشد می‌توان از یک تبدیل‌گر به نام 6LoWPAN^۲ استفاده

^۱ Average Sleep Ratio

^۲ مخفف IPv6 over Low Power Wireless Personal Area Networks

نمود که وظیفه‌ی آن فشرده کردن سر بار^۱ بسته‌های ارسالی با استفاده از این پروتکل است تا با سرباری فشرده‌تر امکان انتقال این بسته‌ها بر بستر یک شبکه‌ی مبتنی بر IEEE 802.15.4 وجود داشته باشد. اما ما به دلیل آن که این مبحث در چهارچوب کلی بحثمان نیست از تشریح بیشتر آن صرف‌نظر کرده و به همین تعریف کوتاه اکتفا می‌نماییم.

۷.۲ IPv۶

برای تعریف این مفهوم ابتدا لازم است به طور کلی در ارتباط با پروتکل آدرس‌دهی به بحث پردازیم. وظیفه‌ی اصلی پروتکل‌های آدرس‌دهی تخصیص یک شناسه‌ی یکتا به دستگاه‌های موجود در یک شبکه است که به کمک این شناسه قابلیت قابلیت شناسایی دستگاه‌ها و همچنین مکان‌یابی آن‌ها وجود داشته باشد. در همین زمینه، ابتدا IPv4 معرفی شد که پس از مدتی در سال ۱۹۹۰ و با افزایش سرعت فراگیر شدن اینترنت، دانشمندان حوزه‌ی شبکه به این حقیقت پی بردند که میزان شناسه‌های یکتا برای شبکه‌ی اینترنت بسیار بیشتر از آن چیز است که IPv4 ارائه می‌دهد. در همین راستا بود که IPv6 ابداع شد.

IPv6 بر خلاف IPv4 که ساختاری ۳۲ بیتی داشت، ساختاری ۱۲۸ بیتی دارد. به این معنا که در تئوری توانایی تولید 2^{128} شناسه یکتا را دارد. این رقم چیزی در حدود 3.4×10^{38} شناسه می‌شود که می‌توان گفت حدودا 7.9×10^{28} برابر تعداد شناسه‌های قابل تولید توسط IPv4 می‌باشد.

اما صرف‌نظر از برتری IPv6 از لحاظ تعداد بیشتر شناسه‌های یکتای تولیدی نسبت به IPv4، یک برتری دیگر نیز در این نسخه از پروتکل آدرس‌دهی مذکور وجود دارد و آن هم ویژگی خودپیکربندی است. به واسطه این ویژگی جدید، دستگاه‌هایی که به شبکه افزوده می‌شوند، برای دریافت شناسه‌ی یکتا (یا همان IP) نیازی به ارتباط با یک سرور DHCP^۲ ندارند. واضح است که این ویژگی تا چه میزان می‌تواند در سیستم‌های مبتنی بر اینترنت اشیا کاربردی باشد؛ چراکه در این سیستم‌ها به طور معمول با حجم عظیمی از تجهیزات و حسگرها مواجه هستیم که نیازمند پیکربندی می‌باشند. همین امر مهم می‌تواند IPv6 را به یکی از راهکاری اصلی برای پروتکل آدرس‌دهی در سیستم‌های مبتنی بر اینترنت اشیا تبدیل نماید.

^۱Overhead

^۲مخفف Dynamic Host Configuration Protocol

حال که به طور اجمالی با عبارات، ابزارها و فناوری‌های مورد استفاده در مقالات تحت بررسی آشنایی یافتیم؛ در فصل بعد به بیان ایده‌های مطرح شده در این مقالات می‌پردازیم. شایان ذکر است که در این بخش، سعی شد تا به حداکثر مباحث پیش‌نیاز پرداخته شود. اما ممکن است برخی اصطلاحات به کار رفته در فصول بعدی، در این بخش مورد بحث واقع نشده باشد؛ که در این صورت در همان فصول و به اختصار به شرح و مرور آن‌ها خواهیم پرداخت.

فصل ۳

کارهای مرتبط

در این بخش به بیان ایده‌های مطرح شده در مقاله‌های تحت بررسی می‌پردازیم. شکل ۱.۳ نمایان‌گر درخت موضوعی تحقیق است و همان‌طور که در تصویر مشاهده می‌نمایید؛ اساساً راهکارهای ارائه شده برای کاهش مصرف انرژی در سیستم‌های مبتنی بر اینترنت اشیا، به دو دسته‌ی کلی راهکارهای نرم‌افزاری و سخت‌افزاری تقسیم می‌گردد. در راهکارهای نرم‌افزاری، دو ایده‌ی اصلی مطرح بوده است که این ایده‌ها به ترتیب هدفشان بهیمنگی ارتباطات میان اجزای سیستم به منظور کاهش اتلاف انرژی در جریان انتقال داده‌ها و دیگری، غیرفعال نمودن اجزا (در این جا بیشتر منظور حسگرها می‌باشند). در شرایط غیرضروری جهت کاهش مصرف انرژی توسط آن‌ها است.

برای رویکرد، اول در برخی مقالات، روش‌های مبتنی بر گراف معرفی شده است که بهیمنگی ارتباطات سیستم را مورد هدف قرار داده‌اند. در ارتباط با رویکرد دوم نیز، معماری‌هایی ارائه شده که در آن‌ها فعال و یا غیرفعال بودن حسگرها بنابر ایده‌ی ابتکاری ارائه شده تعیین می‌گردد. علاوه بر معماری‌های ارائه شده، در برخی مقالات از مکانیزم DRX/DTX برای فعال و غیرفعال نمودن حسگرها استفاده شده است. در ادامه به بررسی و تشریح هر یک از راهکارهای زیر خواهیم پرداخت.

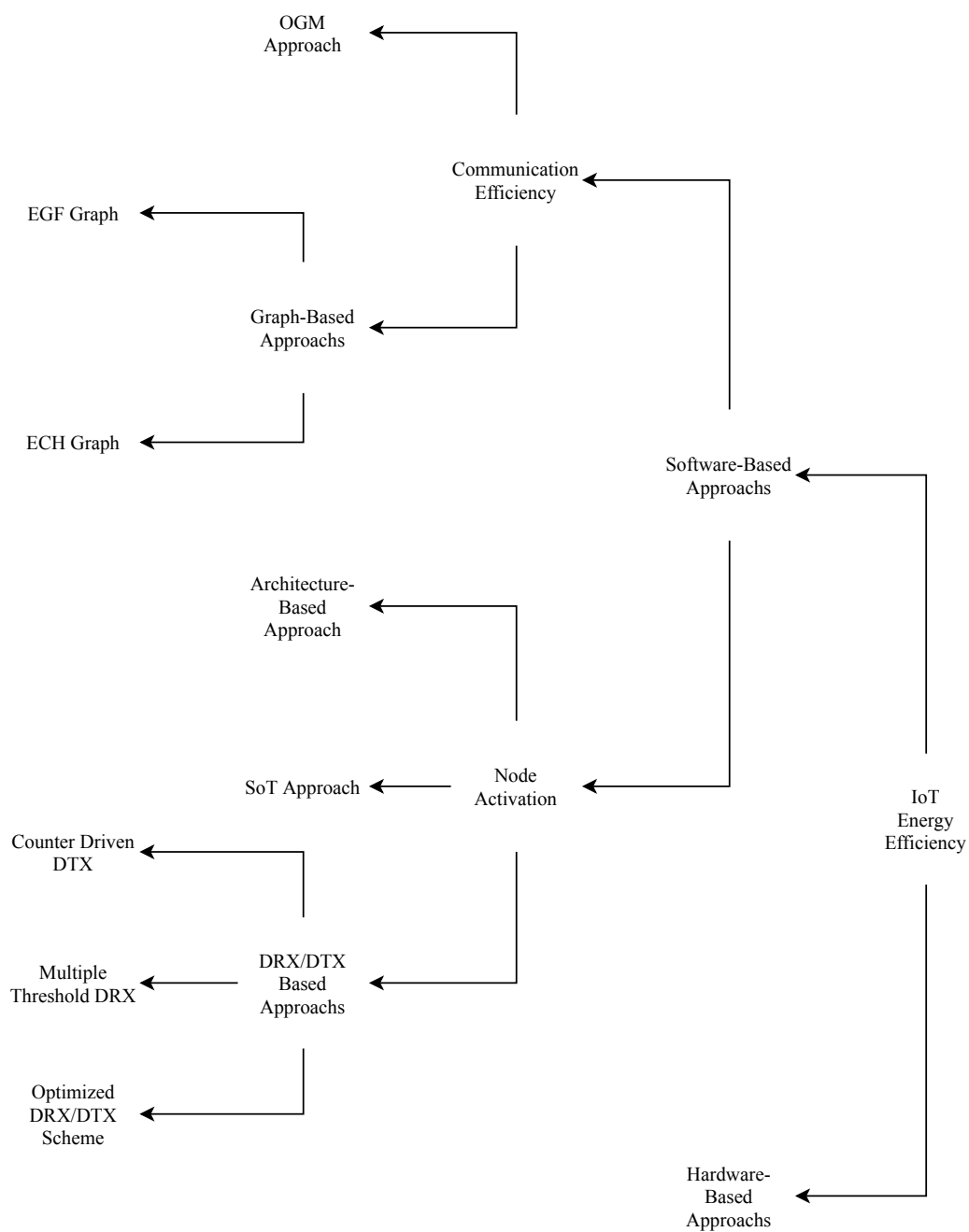
۱.۳ راهکارهای مبتنی بر مکانیزم DRX/DTX

پیشتر در ارتباط با مکانیزم DRX/DTX به تفصیل صحبت کرده‌ایم. حال می‌خواهیم مقالاتی را که از این مکانیزم در بیان راهکارهایشان استفاده کرده‌اند، مورد بررسی قرار دهیم.

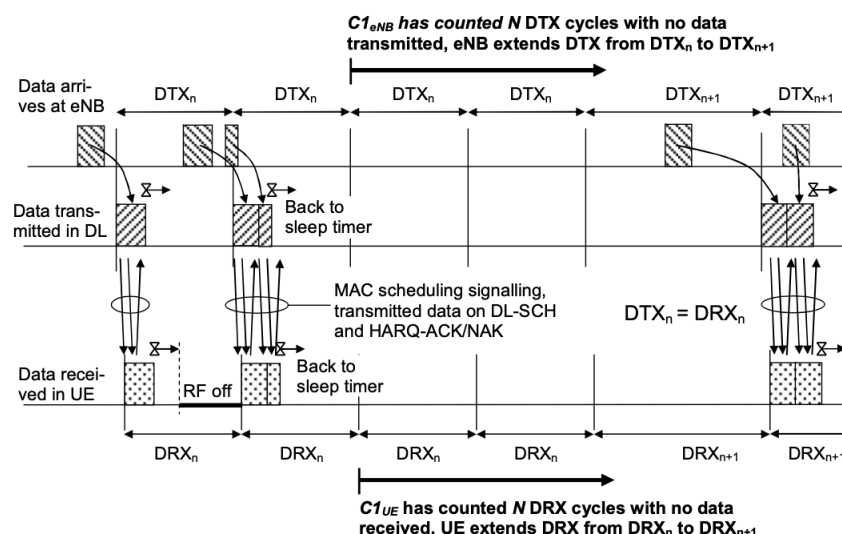
۱.۱.۳ Counter Driven DRX (CDD)

اولین راهکاری که مورد بررسی قرار می‌دهیم، راهکاری است که در مقاله [۱] مطرح گردیده. همان‌طور که پیشتر نیز گفتیم، یک Trade-Off مابین مصرف انرژی و دقت سیستم وجود دارد که می‌توان گفت این مقاله سعی در جهت ایجاد توازن در این امر را دارد. در این مقاله، هر یک از گره‌ها برای ارتباط با گره دیگر، دارای دو شمارنده هستند. در ادامه نحوه عملکرد هر یک از آن‌ها را تشریح می‌نماییم.

برای بیان نحوه کارکرد این شمارنده‌ها لازم است ابتدا با نحوه عملکرد اجزای سیستم آشنایی یابیم. در یک



شکل ۱.۳: درخت موضوعی تحقیق

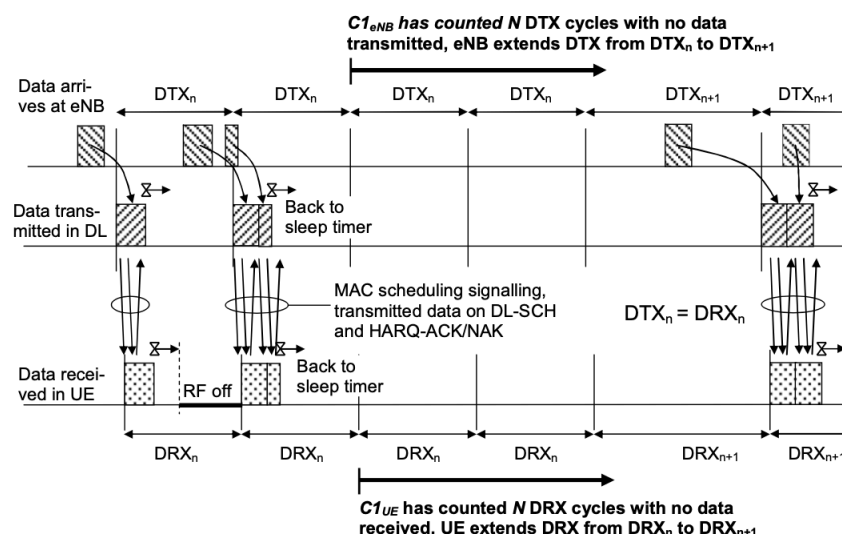


شکل ۲.۳: افزایش دوره DRX در واکنش به کمبود درخواست برای گره

سیستم مبتنی بر DRX/DTX، یک گره در زمان معینی بیدار می‌شود (فعالیت خود را آغاز می‌کند)، در زمانی که فعال است به درخواست‌های رسیده از جانب دیگر گره‌ها پاسخ می‌دهد و پس از پایان زمان فعالیت مجدداً به خواب می‌رود (غیرفعال می‌شود). حال اگر گره، در زمانی از خواب بیدار شود و در تمام طول مدتی که فعال است درخواستی از جانب دیگر گره‌ها دریافت ننماید، شمارنده‌ی اول خود را افزایش می‌دهد. اما اگر در یکی از زمان‌های فعالیت، گره درخواستی از دریافت نماید این شمارنده به مقدار اولیه (یا همان صفر) Reset می‌گردد. پس از آن که مقدار این شمارنده از یک حد حساب^۱ عبور کرد؛ زمان دوره DRX افزایش می‌یابد. همان‌طور که در فصل قبل نیز به آن اشاره شد، با افزایش زمان دوره در این مکانیزم، در واقع زمان فیرفعال بودن گره افزایش می‌یابد. در نتیجه این سیستم در واکنش به کمبود درخواست برای یک گره (یا همان حسگر)، فعالیت آن را کاهش داده تا در مصرف انرژی صرفه‌جویی نماید. در شکل ۲.۳ نمونه فعالیت این راهکار را برای افزایش زمان غیرفعال بودن یک گره مشاهده می‌کنید.

تا این جا رویکرد راهکار ارائه شده در واکنش به عدم نیاز به جمع‌آوری داده از یک حسگر را دیدیم. حال به بررسی دومین شمارنده می‌پردازیم. در این شمارنده که به نوعی می‌توان گفت عکس عمل شمارنده‌ی اول را انجام می‌دهد؛ در هر دوره که گره، بیدار می‌شود؛ در صورتی که در زمان بیداری درخواستی جهت پاسخگویی به گره ارسال شده باشد، شمارنده یک واحد افزایش می‌یابد و در صورتی که درخواستی برای گره ارسال نشود، مقدار

^۱Threshold



شکل ۳.۳: کاهش دوره DRX در واکنش به ازدیاد درخواست برای گره

شمارنده به صفر Reset خواهد شد. همانند شمارنده‌ی قبل؛ برای این شمارنده نیز حدحسابی در نظر گرفته شده است که اگر مقدار شمارنده از این حدحساب عبور کند، دوره تناوب گره کاهش می‌یابد. همان‌طور که گفته شد، این عمل برعکس عمل شمارنده‌ی بالاست. یعنی به کمک این شمارنده، سیستم متوجه می‌شود که گره در هر دوره‌ی بیداری درخواستی برای پاسخگویی داشته و در نتیجه، سیستم درمی‌یابد که نیازمندی سیستم به این گره بیش از مقدار فعلی است. پس برای افزایش دقت سیستم لازم است تا گره با تناوب کوتاه‌تری به فعالیت پردازش که البته همان‌طور که پیشتر هم اشاره شده این امر موجب افزایش انرژی مصرفی گره و در ازای آن، کاهش طول عمر سیستم می‌شود. در شکل ۳.۳ نمونه فرآیند کاهش طول دوره برای یک گره را مشاهده می‌نمایید.

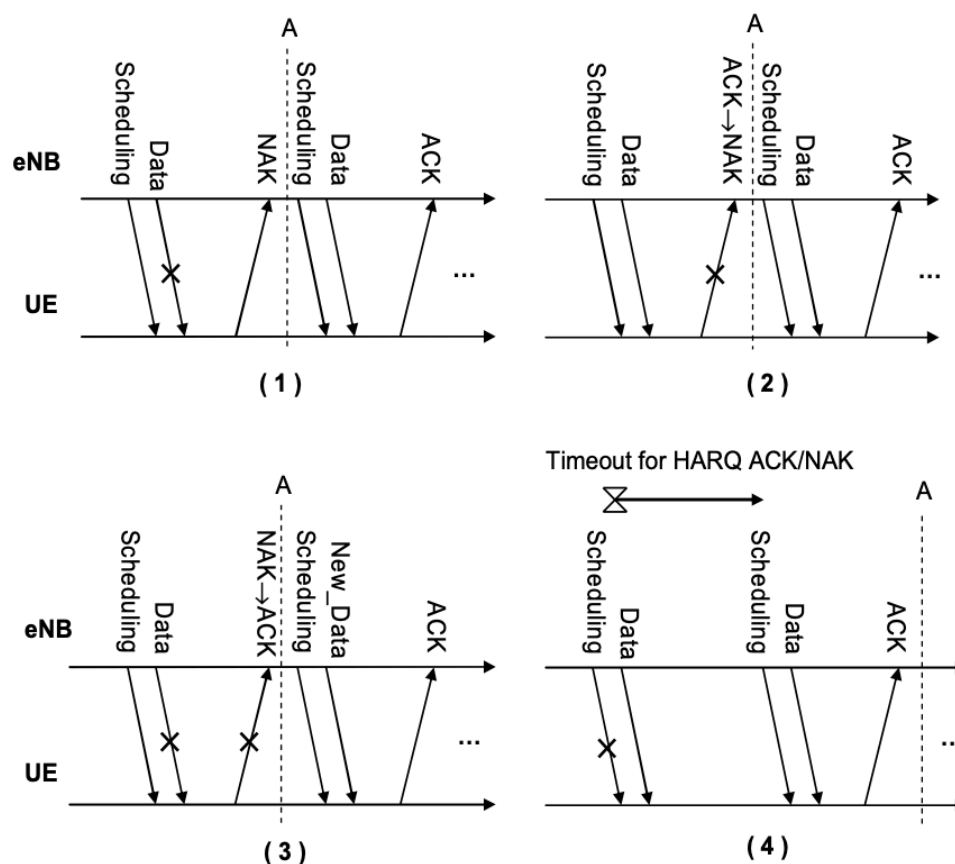
نیرومندی راهکار ارائه شده

منظور از نیرومندی^۱ راهکار ارائه شده، اثبات عملکرد آن در ازای بروز هرگونه خطا می‌باشد. به همین منظور، در مقاله مذکور مجموعاً چهار نوع خطا در حین کارکرد سیستم ارزیابی شده است. که به شرح ذیل می‌باشند.

۱. گره درخواست‌کننده، درخواست را زمان‌بندی می‌کند. اما در ارسال موفق نمی‌شود. گره پاسخگو، سیگنال منفی (NAK) در پاسخ می‌فرستد.

۲. گره درخواست‌کننده، درخواست را زمان‌بندی و ارسال می‌کند. گره پاسخگو، سیگنال را با خطا می‌فرستد.

^۱Robustness



شکل ۴.۳: خطاهای ممکن در حین برقرار ارتباط میان دو گره

۳. گره درخواست کننده، برای ارسال درخواست دچار خطا می شود و گره پاسخگو نیز در ارسال سیگنال پاسخ دچار خطا می گردد.

۴. گره درخواست کننده، در زمان بندی و ارسال درخواست دچار خطا می شود و طبیعتاً به دلیل خطای زمان بندی، امکان ارسال سیگنال از جانب گره پاسخگو وجود ندارد.

خطاهای ذکر شده در بالا را می توانید در شکل ۴.۳ مشاهده نمایید. همانگونه که در تصویر نیز مشهود است؛ در صورت بروز هر یک از خطا دو سیستم از نقطه A به بعد مجدداً با هم همگام شده و عملکرد طبیعی سیستم مشاهده می گردد.

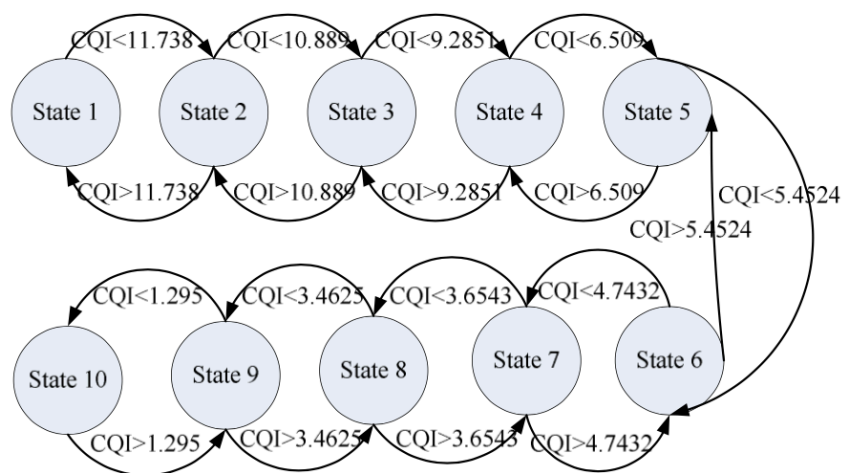
۲.۱.۳ Multiple Threshold DRX (MTD)

راهکار ارائه شده در این مقاله [۱۱]، این است که برخلاف راهکار ارائه شده در مقاله‌ی قبل، تغییری در طول دوره‌ی گره ایجاد نمی‌شود؛ بلکه بر اساس شیوه‌ای به تغییر مقدار زمان Inactivity Timer گره می‌پردازد. در مورد این پارامتر نیز در فصل پیش به تفصیل صحبت کردیم اما برای یادآوری، این پارامتر بیانگر مدت زمانی بود که گره، پس از دریافت درخواست از جانب سایر گره‌ها روشن می‌ماند تا درخواست رسیده را پاسخ دهد و در این مدت به سایر درخواست‌های رسیده پاسخگو نبود.

در راهکار ارائه شده در مقاله مورد بررسی، ۱۰ حالت برای کیفیت عملکرد گره‌ها در نظر گرفته شده است که از حالت ۱ الی ۱۰ زمان Inactivity Timer آن‌ها افزوده شده و در نتیجه کیفیت عملکرد آن‌ها کاهش می‌یابد. دلیل کاهش کیفیت عملکرد از دیدگاه این مقاله، این است که با افزایش زمان Inactivity Timer، امکان دریافت درخواست‌های جدید از گره سلب می‌شود و دقت کاهش می‌یابد. پس در نتیجه، حالت ۱ باکیفیت‌ترین و حالت ۱۰ کم‌کیفیت‌ترین حالت است. این حالات را می‌توانید در جدول ۱.۳ مشاهده نمایید. در ابتدا، تمامی گره‌ها در حالت ۱ قرار دارند. پس از آن، به طور متناوب، شاخص کیفیت (CQI)^۱ هر یک از گره‌ها محاسبه می‌شود. پس از محاسبه‌ی CQI هر یک از حالات مذکور دارای یک حدحساب CQI است که این حدحساب‌ها در شکل ۵.۳ قابل مشاهده است. اگر در مدت‌زمان T_{trig} که در جدول ۱.۳ آمده است، مقدار CQI پایین‌تر از حدحساب باشد، گره از حالت فعلی به یک حالت بالاتر می‌رود و در نتیجه تایمر طولانی‌تر شده که منجر به کاهش کیفیت عملکرد و البته، افزایش طول عمر گره می‌گردد.

برعکس همین عمل نیز در سیستم اتفاق می‌افتد. یعنی اگر برای یک گره، در ازای مدت‌زمان بیشتری از T_{trig} میزان CQI بالاتر از حدحساب حالت کمتر (باکیفیت‌تر) باشد، حالت کیفیت گره به حالت باکیفیت‌تر تقلیل می‌یابد که موجب افزایش دقت عملکرد حسگر می‌گردد.

^۱مخفف Channel Quality Indicator



شکل ۵.۳: حالت‌های کیفیت در MTD و حد حساب هر یک

جدول ۱.۳: مقادیر تایمر مربوط به هر یک از حالات

پارامتر	(TTI) Timer Inactivity DRX
حالت ۱	4
حالت ۲	5
حالت ۳	10
حالت ۴	20
حالت ۵	30
حالت ۶	40
حالت ۷	50
حالت ۸	60
حالت ۹	80
حالت ۱۰	100
T_trig	6

۳.۱.۳ DRX/DTX Optimized

راهکار ارائه شده در مقاله [۱۱]، از بسیاری جهات از راهکارهای ارائه شده در دو مقاله‌ی اول در زمینه استفاده از DRX/DTX برای کاهش مصرف انرژی، بهینه‌تر است. در این مقاله، راهکار ارائه شده به این صورت است که ابتدا، بر اساس مدت زمان قابل قبول (بودجه زمانی) جریان^۱های داده مرتبط با یک گره، دوره‌ی کوتاه آن و سپس دوره‌ی بلندمدت آن را تعیین می‌نماید. سپس بر اساس پارمترهای تعیین شده، مقادیر On Duration و Inactivity Timer به دست می‌آید.

تعیین زمان دوره

در اولین گام از این راهکار، همان‌طور که گفته شد، به تعیین مقدار زمان دوره‌ی کوتاه (یا همان Short Cycle) می‌پردازیم. بر این اساس، ابتدا باید اضطراری‌ترین جریان داده را مشخص نماییم. اضطراری‌ترین جریان داده، جریانی است که بودجه زمانی کمتری داشته و به عبارت دیگر انتظار می‌رود تا در زمان کمتری از جانب گره پاسخ یابد. این پارمتر را برای گره‌ی i با نماد D_i^{min} نمایش می‌دهیم. مدت زمان دوره‌ی کوتاه به وسیله‌ی فرمول ۱.۳ محاسبه می‌شود.

$$T_i^S = \left\lfloor \frac{D_i^{min}}{T_{i-1}^S} \right\rfloor \cdot T_{i-1}^S \quad (۱.۳)$$

در فرمول بالا همان‌طور که مشاهده می‌شود، زمان به دست آمده از D_i^{min} کوچکتر و یا مساوی با آن است. پس حتی امکان پاسخ‌دهی جریان داده‌ای با کمترین زمان قابل قبول را نیز ارضا می‌کند و هیچ جریان داده‌ای به واسطه طولانی بودن زمان دوره از دست نمی‌رود (Miss نمی‌شود). حال، برای به دست آوردن مدت زمان دوره‌ی طولانی مدت، ابتدا در بین سرویس‌های مرتبط با گره، کوتاه‌ترین زمان پاسخگویی قابل قبول را یافته و بر اساس فرمول ۲.۳، محاسبه‌ی زمان دوره را انجام می‌دهیم. کوتاه‌ترین زمان پاسخگویی قابل قبول را برای گره i با نماد S_i^{min} نمایش می‌دهیم.

^۱Flow

$$T_i^L = \left\lfloor \frac{S_i^{min}}{T_i^S} \right\rfloor . T_i^S \quad (۲.۳)$$

همانطور که در فرمول فوق مشاهده می شود، مدت زمان دوره ی طولانی یک گره، در واقع ضربی از مدت زمان دوره ی کوتاه آن می باشد. علاوه بر این، گزاره ی بالایی در مورد این فرمول نیز کاربرد دارد. یعنی به دلیل آن که T_i^L کوچکتر و یا مساوی S_i^{min} است، در نتیجه حتی کوتاه ترین زمان در میان زمان های قابل قبول برای پاسخ دهی به سرویس های مرتبط با گره را نیز ارضا خواهد کرد و به عبارت دیگر، سرویسی به دلیل طولانی بودن زمان دوره از دست نخواهد رفت.

تعیین مدت زمان On Duration و Inactivity Timer

برای محاسبه ی مدت زمان On Duration راهکار ارائه شده به این صورت است که مجموع بیشینه ی حجم بسته ی جریان هایی را در نظر می گیریم که بودجه ی زمانی شان، برابر با مدت زمان دوره ی کوتاه که در مرحله ی قبل محاسبه کردیم باشد. سپس این مقدار را بر حداقل میزان کانال^۱ گره ی مدنظر تقسیم می نماییم. لازم به ذکر است که مقدار از ضرب Ω که بیانگر تعداد بلوک های منبع^۲ است در C_i^{min} که بیانگر حداقل نرخ کانال^۳ بوده و واحد آن نیز (bits/RB) می باشد، محاسبه می شود. علاوه بر این، همان طور که در فرمول ۳.۳ مشاهده می شود؛ یک مقدار حداقلی (در اینجا ۱) نیز برای محاسبه زمان On Duration در نظر گرفته می شود.

$$O_i = \max \left\{ \left\lceil \frac{\sum_{D_j=T_i^S, \forall flow_j \in UE_i} Q_j^{max}}{C_i^{min} \times \Omega} \right\rceil, 1 \right\} \quad (۳.۳)$$

دلیل چنین رویکردی این است که مطمئن باشیم حتی با کمترین منابع، در مدت زمان On Duration قادر به دریافت تمام بسته های جریان داده هستیم.

اما برای محاسبه Inactivity Timer، ابتدا لازم است تا expected packet loss rate را که آن را با نماد $E_{i,j}$ نمایش می دهیم، برای هر گره i و هر جریان داده ای j محاسبه کنیم. با فرض این که تعداد بسته های مربوط به

^۱Channel

^۲Resource Block (RB)

^۳Channel Rate

یک جریان داده، M_j عدد باشد و هر کدام از این بسته‌ها دارای تاخیری در بازه‌ی $1..D_j$ باشد، (اگر بیش از D_j یا همان بودجه‌ی زمانی تاخیر داشته باشیم اصولاً بسته از دست رفته است) مقدار $E_{i,j}$ از فرمول ۴.۳ محاسبه می‌شود.

$$E_{i,j} = \sum_{t_m=1..D_j, m}^{M_j} Prob(t_m) \times Loss(m) \quad (۴.۳)$$

$$Loss(m) = \frac{M_j - (\sum_{m=1..M_j} (\phi_m + \eta_m))}{M_j} \quad (۵.۳)$$

$$\phi_m = \begin{cases} 1, & \text{if } X_m \leq D_j \\ 0, & \text{otherwise} \end{cases} \quad (۶.۳)$$

$$\eta_m = \begin{cases} 1, & \text{if } X_m > D_j \text{ and } Y_m \leq \hat{\Gamma}_j^I \\ 0, & \text{otherwise} \end{cases} \quad (۷.۳)$$

تابع $Prob$ به صورت شهودی از سیستم به دست می‌آید و فرمول $Loss$ در ۵.۳ آمده است. برای محاسبه این تابع برای بسته‌ی m ، به دو پارامتر نیاز است که این پارامترها به ترتیب ϕ_m و η_m می‌باشند. پارامتر اول که در فرمول ۶.۳ آمده است، بیانگر این است که آیا بسته در زمان On Duration گره قابل دریافت است یا خیر و پارامتر دوم که در فرمول ۷.۳ به آن اشاره گردیده است، بیانگر این امر است که آیا بسته در زمان Inactivity Timer قابل دریافت است یا خیر. مجموع این دو پارامتر اگر برابر با یک باشد به این معنی است که بسته از جانب گره قابل دریافت بوده است. همان‌طور که مشاهده می‌شود، در فرمول ۵.۳ از مجموع این دو پارامتر استفاده گردیده است تا احتمال دریافت بسته سنجیده شود.

پس از محاسبه‌ی تمامی $E_{i,j}$ ها، برای هر جریان داده‌ی j مقدار کمینه $E_{i,j}$ به عنوان مقدار کاندید Inactivity Timer انتخاب می‌شود. لازم به ذکر است که دلیل انتخاب مقدار کمینه، این است که مقدار $P_{i,j}^{loss}$ (این پارامتر،

بیانگر میزان قابل قبول Loss شدن بسته‌های جریان j در گره i می‌باشد) ثابت بوده و در نتیجه تاثیری در Loss شدن بسته‌ها ندارد؛ پس بدیهی است که کمترین مقدار Inactivity Timer، به دلیل کمتر بیدار بودن گره، باعث کاهش مصرف انرژی می‌شود. سپس از بین تمامی کاندیدهای انتخاب شده، بیشترین مقدار به عنوان Inactivity Timer گره موردنظر انتخاب می‌گردد. دلیل انتخاب مقدار بیشینه در این مرحله، این است که مقدار انتخاب شده به ازای تمامی جریان‌های داده قابل قبول باشد و به عبارت دیگر، مقدار نهایی Inactivity Timer، $P_{i,j}^{loss}$ مرتبط با همه‌ی جریان‌های داده گره را ارضا کند.

زمان‌بندی بسته‌ها

علاوه بر استفاده از مکانیزم DRX/DTX در مقاله‌ی مورد بررسی، یکی دیگر از ایده‌های مطرح شده در این مقاله، استفاده از یک مکانیزم ابداعی جهت زمان‌بندی بسته‌ها می‌باشد. در این مکانیزم بسته‌های مرتبط با هر گره، در یک صف در گره کنترل‌کننده نگه‌داری می‌شود. سپس گره‌کننده، بلوک منبعی را به هر یک از گره‌هایی که بسته‌ای جهت پردازش داشته باشند و زمان Inactivity Timer آن‌ها رو به اتمام باشد؛ تخصیص می‌دهد. سپس بلوک‌های باقی‌مانده مابین گره‌های دیگر که اولویت پایین‌تری دارند تقسیم می‌شود. این عمل منجر به افزایش کیفیت سیستم شده و احتمال Loss شدن بسته‌های داده‌ای جریان‌های گره را کاهش می‌دهد.

۲.۳ راهکارهای مبتنی بر گراف

در مقالاتی که در ادامه به بررسی آن‌ها خواهیم پرداخت، ایده‌ی اصلی مطرح شده، استفاده از گراف‌های درختی برای کاهش فواصل میان گره‌های شبکه می‌باشد. این کاهش فاصله منجر به کاهش مصرف انرژی به هنگام ارسال و دریافت اطلاعات میان گره‌ها می‌شود. در ادامه به بررسی راهکار مقاله‌ی اول از این دسته خواهیم پرداخت.

۱.۲.۳ درخت EGF

در مقاله [۱] راهکاری که ارائه شده، این است که محیط سیستم مبتنی بر اینترنت اشیا به لحاظ جغرافیایی به نواحی‌ای از محیط‌های محلی خوشه‌بندی شود که در هر یک از آن‌ها یک گره به عنوان گره سرگروه فعالیت

می‌کند. وظیفه‌ی گره سرگروه این است که ارتباط لازم میان گره هدف و Base Station را برقرار نماید. به این وسیله مصرف انرژی شبکه مبتنی بر اینترنت اشیا کاهش چشمگیری می‌یابد. زیرا در هنگام برقراری ارتباط، نیازی به روشن بودن مدار ارتباطی همه گره‌ها نیست و نکته‌ی اصلی همین است که بخش زیادی از انرژی مصرف شده توسط گره‌ها مربوط به مدار ارتباطی آن‌ها می‌باشد.

مدل انرژی

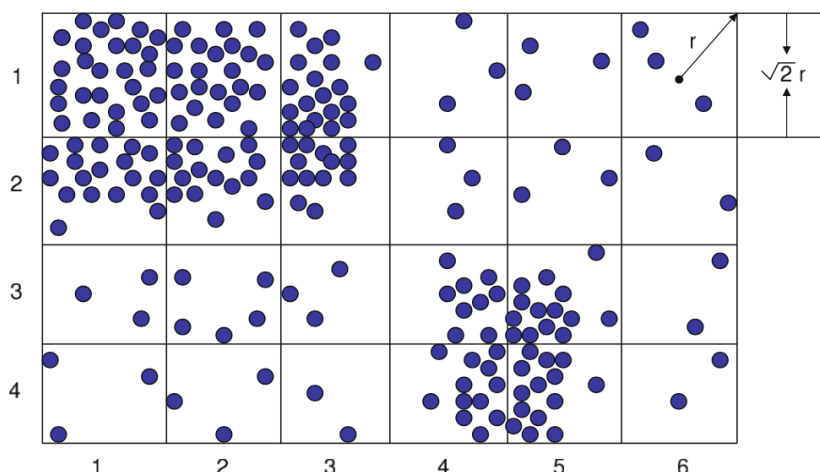
در مقاله‌ی مذکور برای مدل‌سازی ریاضی انرژی از توابع زیر استفاده شده است. لازم به ذکر است که فرمول ۸.۳ بیانگر میزان انرژی لازم برای ارسال k بیت پیام به فاصله‌ی d ، فرمول ۹.۳ بیانگر میزان انرژی مصرفی جهت دریافت k بیت پیام و فرمول ۱۰.۳ بیانگر میزان انرژی لازم برای انتقال (ارسال و دریافت) k بیت پیام به فاصله d می‌باشد.

$$E_{tx} = a \times k + b \times k \times d_{ij}^n \quad (۸.۳)$$

$$E_{rx} = c \times k \quad (۹.۳)$$

$$C_{ij}(K) = \begin{cases} a \times k + b \times k \times d_{ij}^n & \text{if } j \text{ is } BS \\ E_{tx} + E_{rx} & \text{otherwise} \end{cases} \quad (۱۰.۳)$$

همان‌گونه که در فرمول‌های بالا ملاحظه می‌شود؛ a ، b و c ثابت هستند و d_{ij} بیانگر فاصله‌ی میان دو گره i و j است. علاوه بر این، این نکته از فرمول‌های بالا برداشت می‌شود که دریافت پیام توسط BS یا همان Base Station محدودیتی ندارد و به همین دلیل در مدل ریاضی مقاله‌ی تحت بررسی از این مقدار چشم‌پوشی شده است.



شکل ۶.۳: محیط مورد بررسی در مقاله‌ی [۱]

ساخت درخت EFG

فرض کنید محیطی که قرار است درخت را در آن تشکیل دهیم، محیط نمایش داده شده در شکل ۶.۳ است. برای ساخت درخت، همانطور که در الگوریتم شکل ۷.۳ دیده می‌شود، ابتدا محیط را به سلول‌های مربعی با طول ضلع $\sqrt{2}r$ (که r بیانگر شعاع ارتباطی حسگرها است و در این مقاله برای تمامی حسگرها یکسان است) تقسیم‌بندی می‌نماییم.

پس از آن که تقسیم‌بندی سلول‌ها انجام گردید؛ در الگوریتم ارائه شده، نوبت به مرحله‌ی محاسبه‌ی وزن می‌رسد. در این مرحله، به ازای هر گره، وزن میان این گره و گره‌های مجاور آن (یعنی گره‌های بالا، پایین، چپ و راست گره مورد نظر) به کمک الگوریتم دوم که در شکل ۸.۳ مشهود است؛ محاسبه می‌شود. در ادامه به توضیح این الگوریتم خواهیم پرداخت.

اما بعد از آن که محاسبه‌ی وزن میان سلول‌های محیط صورت گرفت؛ ابتدا تمامی سلول‌ها را به عنوان یک زیرمحیط^۲ در نظر می‌گیریم و وزن میان هر جفت از این زیرمحیط‌ها را به صورت نزولی مرتب می‌نماییم. در این مرحله، هر بار؛ دو زیرمحیط که بین‌شان بالاترین وزن موجود است با یکدیگر ادغام کرده و یک زیرمحیط جدید ایجاد می‌کنیم که در واقع والد دو زیرمحیط قبلی است. بدین صورت گره‌های دو زیرمحیط قبلی به عنوان گره‌های

^۱Cell

^۲Sub-Region

فرزند^۱ زیرمحیط جدید معرفی می‌شوند.

پس از آنکه زیرمحیط جدید تعریف شد، در واقع همان مرحله‌ی قبلی - که محاسبه وزن بود - باید برای آن صورت گیرد. اما با این تفاوت که این بار، وزن باید برای یک زیرمحیط محاسبه شود و نه یک سلول که روش محاسبه‌ی وزن برای زیرمحیط در الگوریتم شکل ۹.۳ آمده است. در مورد این الگوریتم نیز جلوتر به تفصیل صحبت می‌نماییم. سپس، با محاسبه‌ی وزن زیرمحیط‌های مجاور زیرمحیط مدنظر (زیرمحیط‌های بالا، پایین، چپ و راست آن) کار الگوریتم در گام اول به پایان می‌رسد و عملیات ذکر شده در دو پاراگراف اخیر (خط‌های ۱۰ الی ۱۷ شبه کد الگوریتم شکل ۷.۳) تکرار می‌گردد. این عملیات تا آن جایی ادامه پیدا می‌کند که دیگر فقط یک زیرمحیط در تمام محیط تحت پوشش سیستم باقی بماند. به این صورت گراف محیط شکل ۶.۳ ساخته شده و در شکل ۱۰.۳ قابل مشاهده است. لازم به ذکر است که در گراف درختی ساخته شده، سلول‌ها در موقعیت برگ‌های درخت قرار می‌گیرند.

در الگوریتم شکل ۸.۳ برای محاسبه‌ی وزن میان دو سلول مجاور، به ازای هر جفت گره‌ای از سلول اول و سلول دوم (یعنی گره اول در جفت گره موردنظر، متعلق به سلول اول و گره دوم متعلق به سلول دوم باشد) عملیات زیر را انجام می‌دهیم. ابتدا، بررسی انجام می‌شود که فاصله‌ی دو گره از هم بیشتر از شعاع ارتباطی (یا همان r) نباشد. سپس در صورتی که فاصله‌ی بین دو گره، کمتر و یا مساوی شعاع ارتباطی‌شان بود؛ وزن میان آن دو گره از معکوس $C_{ij}(k)$ که محاسبه‌ی آن در فرمول ۱۰.۳ شرح داده شد به دست می‌آید. سپس از مجموع وزن‌های به دست آمده در میان گره‌ها، وزن کلی میان دو سلول حاصل می‌گردد. در فرمول ۱۱.۳ می‌توانید نمونه محاسبه وزن میان دو سلول قابل مشاهده در شکل ۱۱.۳ را بررسی نمایید.

$$w_{(3,1),(3,2)} = \frac{1}{C_{S_2,S_4}(k)} + \frac{1}{C_{S_2,S_5}(k)} + \frac{1}{C_{S_3,S_4}(k)} \quad (11.3)$$

از فرمول بالا می‌توان دریافت که فاصله‌ی دو گره با وزن آن نسبت عکس دارد. یعنی هر آن چه فاصله‌ی میان گره‌ها بیشتر باشد، انرژی مصرفی بیشتری مورد نیاز است و همین باعث کاهش وزن رابطه‌ی میان آن دو گره در گراف می‌شود.

^۱Child Nodes

Algorithm 1 : EGFTreeConstr

Require: SN_{set} : set of sensor nodes

r : communication radius of sensors

Ensure: nd_{rt} : the root node of EGF-tree

- 1: divide the whole region evenly into grid cells where the side length of each grid cell is $\sqrt{2} r$
 - 2: **foreach** grid cell gc **do**
 - 3: $NGS_{gc} \leftarrow$ the grid cells in the up-, down-, left- and right-side grid cells of gc
 - 4: **foreach** $ngc \in NGS_{gc}$ **do**
 - 5: $GrdWtCal(gc, ngc)$ as shown in Algorithm 2
 - 6: **end for**
 - 7: **end for**
 - 8: $num_{sr} \leftarrow$ number of sub-regions in the whole region
 - 9: **while** $num_{sr} > 1$ **do**
 - 10: sort all weighs of any two neighbor sub-regions from big to small
 - 11: new sub-region $nr \leftarrow$ merge two neighbor sub-regions: nr_1 and nr_2 , with the biggest weight
 - 12: set nr_1 and nr_2 as child nodes of nr
 - 13: $NBS_{nr} \leftarrow$ the grid cells in the up-, down-, left- and right-side sub-regions of nr
 - 14: **foreach** $nbr \in NBS_{nr}$ **do**
 - 15: $IntRgWtCal(nr, nbr)$ as shown in Algorithm 3
 - 16: **end for**
 - 17: $num_{sr} \leftarrow num_{sr} - 1$
 - 18: **end while**
 - 19: $nd_{rt} \leftarrow nr$
 - 20: head nodes selection
-

شکل ۷.۳: شبه کد الگوریتم ساخت درخت EGF

Algorithm 2 : GrdWtCal

Require: SN_{set_1}, SN_{set_2} : sets of sensor nodes in two neighbor grid cells: gc_1 and gc_2

Ensure: w_{gc_1, gc_2} : the weight between gc_1 and gc_2 . It is initially set to zero

```

1: foreach  $s_1 \in SN_{set_1}$  do
2:   foreach  $s_2 \in SN_{set_2}$  do
3:     if  $dis(s_1, s_2) \leq r$  then
4:        $C_{s_1, s_2}(k) \leftarrow (a + c) \times k + b \times k \times d_{ij}^n$ 
5:        $w_{s_1, s_2} \leftarrow \frac{1}{C_{s_1, s_2}(k)}$ 
6:        $w_{gc_1, gc_2} \leftarrow w_{gc_1, gc_2} + w_{s_1, s_2}$ 
7:     end if
8:   end for
9: end for

```

شکل ۸.۳: شبه کد الگوریتم محاسبه‌ی وزن میان سلول‌ها

Algorithm 3 : IntRgWtCal

Require: nr_1, nr_2 : two sub-regions nearby

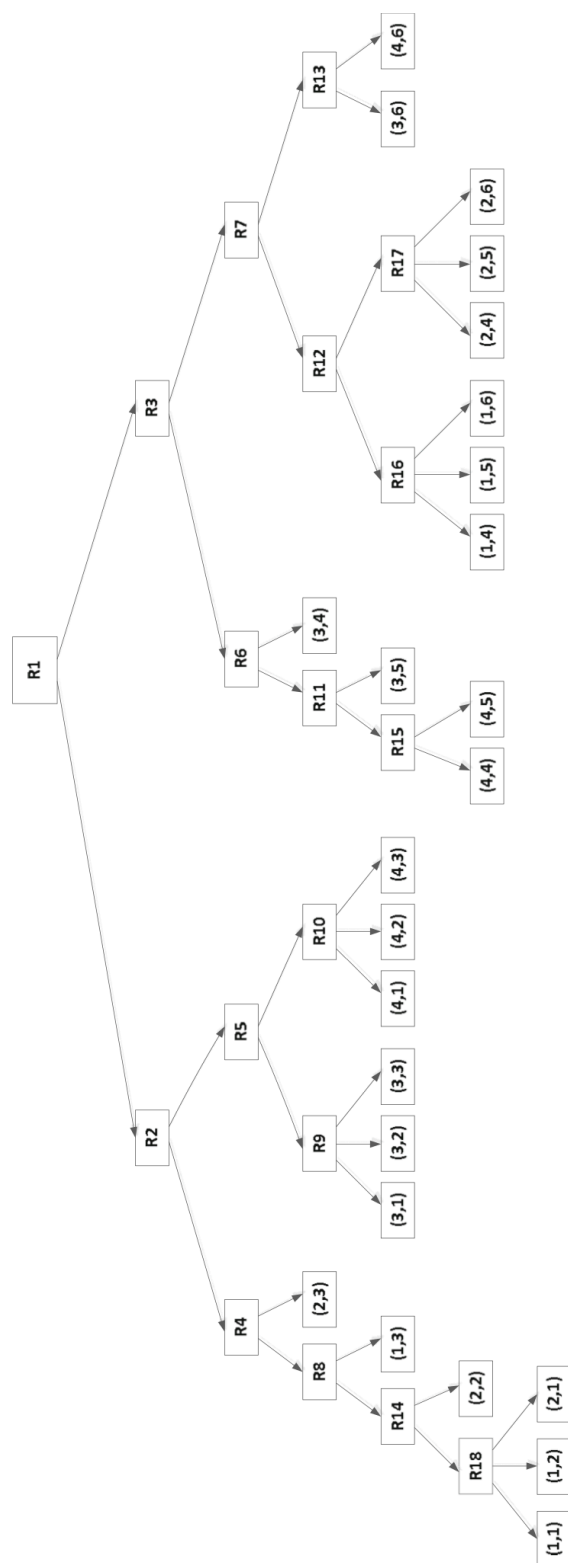
Ensure: w_{nr_1, nr_2} : weight for nr_1 and nr_2

```

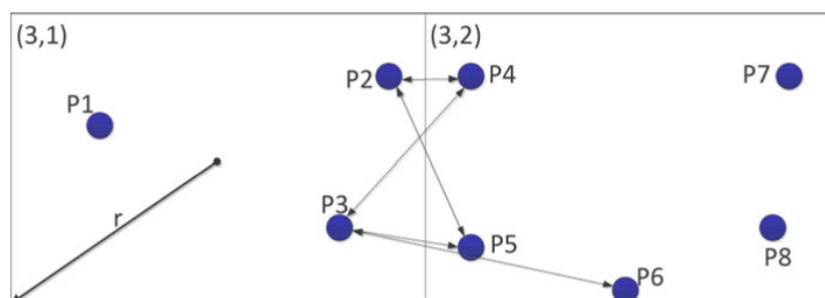
1:  $GC_{set_1} \leftarrow$  grid cells in  $nr_1$ 
2:  $GC_{set_2} \leftarrow$  grid cells in  $nr_2$ 
3: foreach  $gc_1 \in GC_{set_1}$  do
4:   foreach  $gc_2 \in GC_{set_2}$  do
5:      $NBS_{gc_2} \leftarrow$  set of grid cells, where each cell
      is (i) in  $GC_{set_1}$ , and (ii) is on the up-, down-,
      left-, right-side of  $gc_2$ 
6:     if  $gc_1 \in NBS_{gc_2}$  then
7:        $w_{gc_1, gc_2} \leftarrow \text{GrdWtCal}(gc_1, gc_2)$ 
8:        $w_{nr_1, nr_2} \leftarrow w_{nr_1, nr_2} + w_{gc_1, gc_2}$ 
9:     end if
10:   end for
11: end for

```

شکل ۹.۳: شبه کد الگوریتم محاسبه‌ی وزن میان زیرمناطق‌ها



شکل ۱۰.۳: گراف حاصل شده برای محیط شکل ۶.۳



شکل ۱۱.۳: محاسبه‌ی وزن میان دو سلول مجاور

اما برای محاسبه‌ی وزن میان دو زیرمحیط، همان‌گونه که در الگوریتم شکل ۹.۳ مشاهده می‌شود، به این صورت است که برای هر سلول در زیرمحیط دوم، سلول‌هایی را می‌یابیم که اولاً؛ عضو زیرمحیط اول باشند و دوماً؛ با سلول مدنظر مجاورت داشته باشند. سپس باقی مراحل همانند محاسبه وزن برای سلول‌ها خواهد بود. بدین ترتیب که به ازای هر جفت سلولی که با یکدیگر، رابطه‌ی مذکور را داشتند وزن را محاسبه می‌کنیم و از مجموع اوزان به دست آمده، وزن کلی مابین دو زیرمحیط حاصل می‌گردد.

تجمیع کوئری‌ها

یکی از راهکارهای ارائه شده‌ی دیگر در این مقاله، تجمیع کوئری‌ها است. تجمیع کوئری‌ها باعث می‌شود تا درخواست ارسالی به حسگرها از طریق یک رابط منتقل گردد و سپس در برگشت اطلاعات نیز تمامی حسگرها، اطلاعات خود را به یک گره سرگروه ارسال کرده و ادغام این اطلاعات با یکدیگر در گره سرگروه انجام شده و نتیجه فقط از طریق از طریق ارتباط آن گره با Base Station بازگردانی شود. برای این کار، لازم است ابتدا مدل کوئری ارائه شده را تعریف نماییم.

$$Q = \langle ID, R, T, A, F \rangle \quad (۱۲.۳)$$

در فرمول ۱۲.۳ مدل کوئری‌های ارائه شده قابل مشاهده است. در این مدل، ID ، بیانگر شناسه کوئری، R بیانگر محیط مرتبط با کوئری، T مشخص کننده‌ی مدت زمان کوئری، A بیانگر مشخصه‌ها و پارامترهای ارسالی کوئری و F بیانگر تناوب کوئری می‌باشد. نکته‌ی مهم این است که هر کوئری در واقع متشکل از چندین زیرکوئری^۱

^۱Sub-Query

می باشد که در پارامترهای کوئری به غیر از Q یکسانند و به عبارت دیگر، هر زیرکوئری، بیانگر یک کوئری مرتبط با یکی از زیرمحیطهای موردنیاز کوئری اصلی می باشد. در فرمول ۱۳.۳ این مسئله به خوبی مشهود است.

$$Q = \langle R, T, A, F \rangle = \langle R_1, T, A, F \rangle + \langle R_2, T, A, F \rangle + \dots + \langle R_j, T, A, F \rangle \quad (13.3)$$

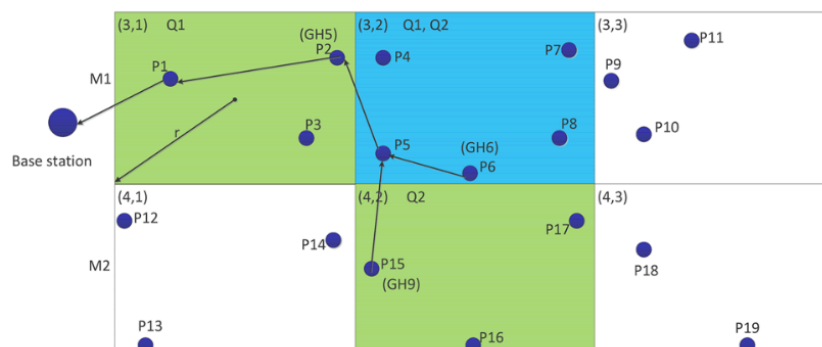
حال که به مفهوم زیرکوئری پرداختیم؛ می توان تجمیع زیرکوئری ها را تشریح کرد. برای تجمیع زیرکوئری ها، دو وضعیت مطرح شده است. به این صورت که یا یک زیرکوئری به لحاظ محیط (R)، زیرمجموعه ی دیگری باشد. و یا دو زیرکوئری به لحاظ محیطشان دارای اشتراک باشند به طوری که رابطه ی $S_{R_i \cap R_j} / S_{R_i \cup R_j} \geq \alpha$ برقرار باشد. (α یک ثابت است که مقدار آن در بازه 0..1 می باشد).

نحوه ی تجمیع زیرکوئری ها را در وضعیت اول و دوم که در بالا ذکر شده است، به ترتیب در فرمول های ۱۴.۳ و ۱۵.۳ آمده است. لازم به ذکر است که در فرمول ۱۴.۳، $R_i \subseteq R_j$ و در فرمول ۱۵.۳، $R_{ij} = R_i \cap R_j$ می باشد.

$$Q_{i+j} = \begin{cases} \langle R_j, \{T_i, T_j\}, A_i, F_i \rangle & \text{if } A_i = A_j \text{ and } F_i = F_j \\ \langle R_j, T_i, \{A_i, A_j\}, F_i \rangle & \text{if } T_i = T_j \text{ and } F_i = F_j \end{cases} \quad (14.3)$$

$$Q_{i+j} = \begin{cases} \langle R_{ij}, \{T_i, T_j\}, A_i, F_i \rangle & \text{if } A_i = A_j \text{ and } F_i = F_j \\ \langle R_{ij}, T_i, \{A_i, A_j\}, F_i \rangle & \text{if } T_i = T_j \text{ and } F_i = F_j \end{cases} \quad (15.3)$$

تا این جا با سازوکار تجمیع داده ها کوئری ها آشنا شدیم. در مرحله بازگشت داده ها در پاسخ به کوئری نیز همین شرایط مطرح است. برای مثال، در شکل ۱۲.۳ می بینیم که $GH5$ به عنوان گره سرگروه انتخاب می شود و داده های به دست یافته از سلول های $(3, 2)$ و $(4, 2)$ ، به وسیله ی سرگروه هایشان ($GH6$ و $GH9$) و از طریق گره $P5$ به $GH5$ انتقال می یابد. داده های به دست آمده از این سه حسگر، در خود $GH5$ تجمیع می شود و چون داده ی دیگری برای تجمیع با داده های فعلی وجود ندارد، نیازی به سطح بالاتری از گره ها نداریم. در نهایت داده های تجمیع شده در $GH5$ از طریق گره $P1$ به Base Station بازگردانده می شوند.



شکل ۱۲.۳: بازگرداندن داده به Base Station

علاوه بر این، در این مقاله برای تعیین گرهی سرگروه هر زیرمحیط از الگوریتم LEACH استفاده شده است که با تغییر مداوم سرگروه‌ها از کاهش چشم‌گیر یک گره خاص جلوگیری نموده و منجر به افزایش طول عمر سیستم می‌گردد.

۲.۲.۳ درخت ECH

از بسیاری جهات، رویکرد مقاله [] همانند مقاله قبلی است. هر دو از رویکرد تجمیع کوثری یکسانی استفاده کرده‌اند، هر دو مدل‌های انرژی و کوثری‌شان یکسان است و هر دو اساس کارشان ساختن درخت برای توپولوژی شبکه مبتنی بر اینترنت اشیا می‌باشد. اما تفاوت عمده این دو رویکرد در نحوه ساخت درختشان است که عملکردشان را متفاوت می‌سازد.

در الگوریتم ارائه شده توسط این مقاله که در شکل ۱۳.۳ دیده می‌شود، همانند مقاله پیشین، ابتدا محیط به سلول‌های با ضلع $\sqrt{2}r$ تقسیم می‌گردد. سپس سلول‌های تقسیم‌بندی شده، هر یک به عنوان یک زیرمحیط در نظر گرفته می‌شوند و الگوریتم شکل ۱۴.۳ بر روی آن‌ها اعمال می‌گردد. می‌توان گفت که اساس کار این مقاله، در واقع همین الگوریتم است و تفاوت اصلی رویکرد این مقاله با مقاله پیشین در همین مرحله رقم می‌خورد. در راهکار ارائه شده در این مقاله، دو عدد ثابت به عنوان حداقل و حداکثر تعداد گره‌های فرزند (و یا در سطوح بالاتر: زیرمحیط‌ها) در یک زیرمحیط در نظر گرفته شده است که به ترتیب، با m و M نمایش داده می‌شوند. الگوریتم شکل ۱۴.۳ بیان می‌دارد که ابتدا تعداد گره‌های فرزند بررسی می‌شود. اگر تعداد گره‌ها در بازه‌ی $m \dots M$ بود، همان گره‌ها به عنوان یک زیرمحیط با سطح بالاتر به عنوان خروجی الگوریتم برگردانده می‌شود. اما اگر چنین

نبود و تعداد بیش از M باشد؛ تعداد k گره از میان گره‌های فرزند انتخاب می‌شود. لازم به ذکر است که چون تعداد گره‌ها برابر با num و تعداد حداقل و حداکثر فرزندان به ترتیب، برابر با m و M می‌باشد؛ پس مقدار k باید در بازه $num/M \dots num/m$ قرار داشته باشد.

پس از آن که این انتخاب در بین گره‌های فرزند صورت گرفت. برای هر گره باقی‌مانده وزن آن نسبت به k گره انتخاب شده سنجیده می‌شود و در این بین هر کدام که وزن بیشتری را به عنوان حاصل، به دست آورد؛ آن گره فرزند گره مذکور از بین k گره انتخابی می‌شود و اگر تعداد زیرگره‌ها برای یک گره بیش از M شود. همین الگوریتم روی آن گره مخصوص مجدداً اعمال می‌گردد. لازم به ذکر است که الگوریتم محاسبه‌ی وزن دقیقاً همانند الگوریتم‌های شکل ۸.۳ و ۹.۳ بوده و این عملیات تا زمانی ادامه می‌یابد که تعداد زیرمحیط‌ها از m بیشتر باشد. به این ترتیب، زمانی که تعداد گره‌ها کمتر از m شد؛ ریشه‌ی درخت مشخص شده و عملیات ساخت درخت پایان می‌یابد.

در پایان می‌توانید درخت ساخته شده توسط الگوریتم ابداعی این مقاله را در شکل ۱۵.۳ مشاهده نمایید. همان‌گونه که در مقایسه‌ی این درخت با درخت شکل ۱۰.۳ مشهود است؛ وجه مشابهت هر دو درخت در این است که در هر دوی آن‌ها، سلول‌های تقسیم‌بندی شده‌ی محیط برگ‌های درخت می‌باشند. اما عمده تفاوت این دو درخت این است که درخت به دست آمده از الگوریتم مقاله‌ی [۱]، درختی متوازن است و عمق کمتری دارد. همین امر می‌تواند سبب کاهش مسیر ارتباطی داده‌ها و درخواست‌ها گردیده و در نتیجه، میزان انرژی مصرفی سیستم‌های مبتنی بر اینترنت اشیا را کاهش دهد.

۳.۳ راهکار (OGM) Object Group Mobility

در راهکار ارائه شده در مقاله‌ی [۱]، که با نام Object Group Mobility معرفی شده است؛ شبکه‌ای از گره‌ها در نظر گرفته شده که می‌توانند در حرکت باشند (موقعیت‌شان را تغییر دهند) و همچنین در محیط مورد بررسی می‌تواند گروه‌های گوناگونی از گره‌ها وجود داشته باشد که فناوری ارتباطی آن‌ها با یکدیگر یکسان نیست. برای مثال در یک گروه ممکن است فناوری ارتباطی IEEE 802.15.4 به کار رود. در صورتی که در گروهی دیگر از RFID و یا Bluetooth استفاده می‌شود. پس به همین دلیل می‌توان گفت شبکه‌ی مورد بررسی و به تبع آن، راهکار ارائه

Algorithm 1. ECHTreeConstr.

Require: SN_{set} : set of sensor nodes
 r : communication radius of sensors
Ensure: nd_{rt} : the root node of ECH-tree

- 1: divide the whole region evenly into grid cells where the side length of each grid cell is $\sqrt{2}r$
- 2: $GC_{set} \leftarrow$ the set of grid cells after grid division
- 3: $CSR_{set} \leftarrow \text{SubRegCluster}(GC_{set})$ as presented by [Algorithm 2](#)
- 4: $num_{csr} \leftarrow$ the number of new formed sub-regions after clustering grid cells
- 5: **While** $num_{csr} \geq m$ **do**
- 6: $CSR_{set} \leftarrow \text{SubRegCluster}(CSR_{set})$ as presented by [Algorithm 2](#)
- 7: $num_{csr} \leftarrow$ the number of new formed sub-regions in CSR_{set}
- 8: **end while**
- 9: $nd_{rt} \leftarrow$ the only new formed cluster region
- 10: head nodes selection

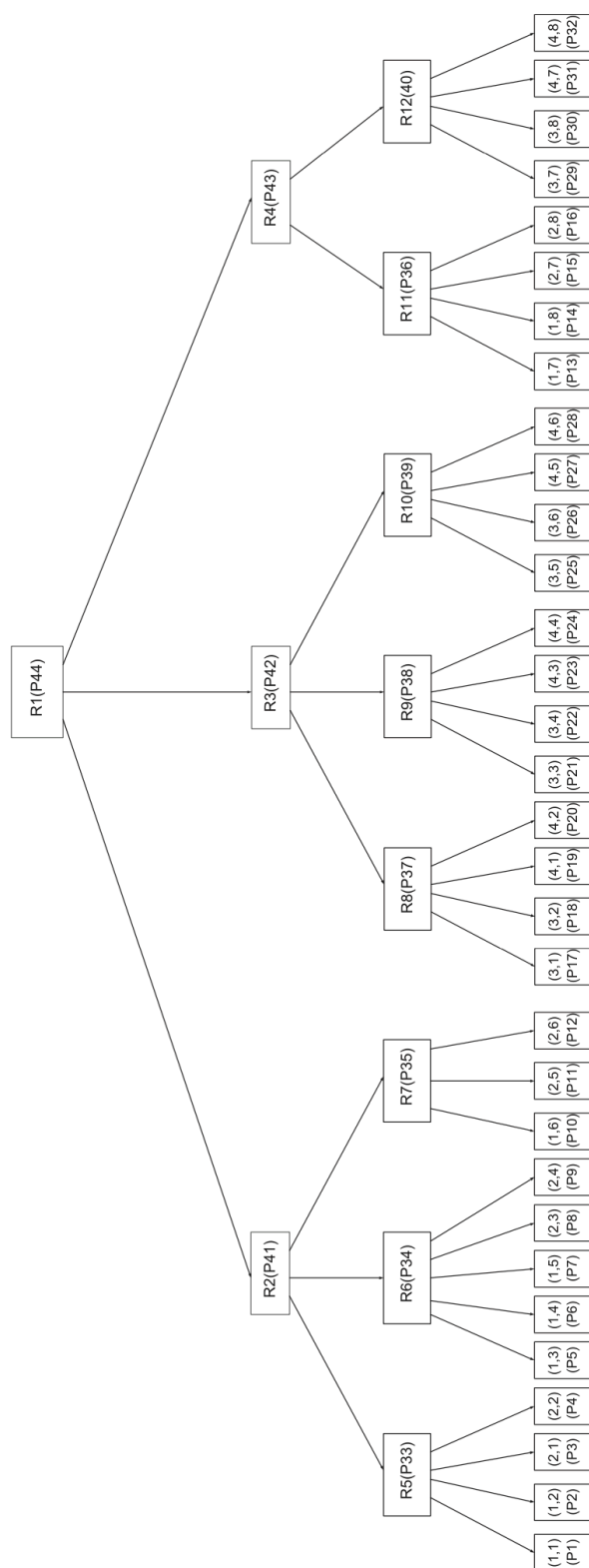
شکل ۱۳.۳: شبه کد الگوریتم ساخت درخت ECH

Algorithm 2. SubRegCluster.

Require: SR_{set} : set of sub-regions in the region
Ensure CSR_{set} : new formed sub-regions after clustering

- 1: $num \leftarrow$ the number of sub-regions in SR_{set}
- 2: **if** $m \leq num \leq M$ **then**
- 3: $CSR_{set} \leftarrow$ sub-regions are clustered into one cluster which is the root node region
- 4: **return**
- 5: **end if**
- 6: $CLC_{sr} \leftarrow$ select k (s.t. $num/M \leq k \leq num/m$) initial sub-regions as cluster centers
- 7: $RAN_{sr} \leftarrow SR_{set} - CLC_{sr}$
- 8: **foreach** $sr_i \in CLC_{sr}$ **do**
- 9: **foreach** $sr_j \in RAN_{sr}$ **do**
- 10: $w_{sr_i, sr_j} \leftarrow$ compute the weight of two sub-regions (or grid cells): sr_i, sr_j , as detailed in [Section 3.2](#)
- 11: **if** $w_{sr_j}^{bg} < w_{sr_i, sr_j}$ **then**
- 12: $w_{sr_j}^{bg} \leftarrow w_{sr_i, sr_j}$
- 13: $CL_{sr_i} \leftarrow sr_j$
- 14: set sr_j as child node of the new formed cluster sub-regions centered at sr_i
- 15: **end if**
- 16: **end for**
- 17: **end for**
- 18: $CSR_{set} \leftarrow$ the new formed cluster sub-regions

شکل ۱۴.۳: شبه کد الگوریتم خوشه‌بندی ECH



شکل ۱۵.۳: گراف ساخته شده با الگوریتم ECH

شده دارای انعطاف بیشتری است.

در این مقاله، فرض شده که محیط مورد بررسی A متشکل از مجموعه ای از محیط‌های کوچک‌تر است که به آن‌ها Ambience گفته می‌شود. ما برای سادگی و یکپارچگی اصطلاحات به کار رفته تا بدین جا، از همان لفظ زیرمحیط برای بیان این مفهوم استفاده می‌نماییم. پس محیط A به شکل $A = \{a_1, a_2, \dots, a_M\}$ قابل بیان است که a_i بیان‌گر زیرمحیط i ام می‌باشد.

سیستم ابداعی این مقاله، در واقع متشکل گروه‌هایی است که خود، مجموعه‌ای از Access Point (AP) های ناهمگن هستند و می‌توانند در کنار یکدیگر کار کنند. برای مثال اگر سیستم مورد بحث ما یک ساختمان هوشمند باشد، هر یک از واحدهای آن یک گروه در سیستم ماست. هر یک از AP های یک گروه، دارای فناوری ارتباطی مخصوص خودش است که با $\psi(AP)$ نمایش داده می‌شود. علاوه بر این، هر AP تعدادی از زیرمحیط‌ها را پوشش می‌دهد که مجموعه‌ی زیرمحیط‌های تحت پوشش آن را با $C(AP)$ نمایش می‌دهند. بدیهی است که هر حسگر فقط می‌تواند به AP هایی وصل شود که فناوری ارتباطی یکسانی با آن داشته باشند.

در راهکار ارائه شده، یک Content Server (CS) به کنترل موقعیت حسگرها می‌پردازد و علاوه بر این، هر حسگر دارای یک Home Network ($HN(n)$) است که در آن یک Agent (که در مقاله با نام Home Agent معرفی شده است) مسئول نگه‌داری اطلاعات مربوط به موقعیت حسگر است. برای این که به سازوکار ابداعی این مقاله آشنا شویم؛ فرض کنید یک گروه از گره‌ها را در اختیار داریم که آن‌ها را با n_0, \dots, n_m نمایش می‌دهیم. فرض می‌نماییم که در این گروه، گره n_0 به عنوان گره سرگروه انتخاب شده است. هر گره پیرو^۱، آدرسی که سرگروه به آن نسبت می‌دهد به عنوان Care-of-Address خود ($CoA(n_i)$) معرفی می‌نماید. این آدرس در واقع همان $HN(n_0)$ است. بدین ترتیب زمانی که یک گره بخواهد با یکی از گره‌های n_1, \dots, n_m ارتباط برقرار کند، درخواست خود را (که می‌تواند شامل داده‌های مورد انتقالی و یا ... باشد) به $CoA(n_i)$ ارسال می‌نماید. سپس براساس $CoA(n_i)$ گره مقصد، درخواست به $HN(n_0)$ منتقل^۲ می‌گردد و از آن جا توسط n_0 برای گره موردنظر ارسال می‌شود.

از راهکار ارائه شده در بالا چنین بر می‌آید که اگر یک گره بخواهد موقعیت خود را تغییر دهد نیازمند به‌روزرسانی

^۱Slave

^۲Forward

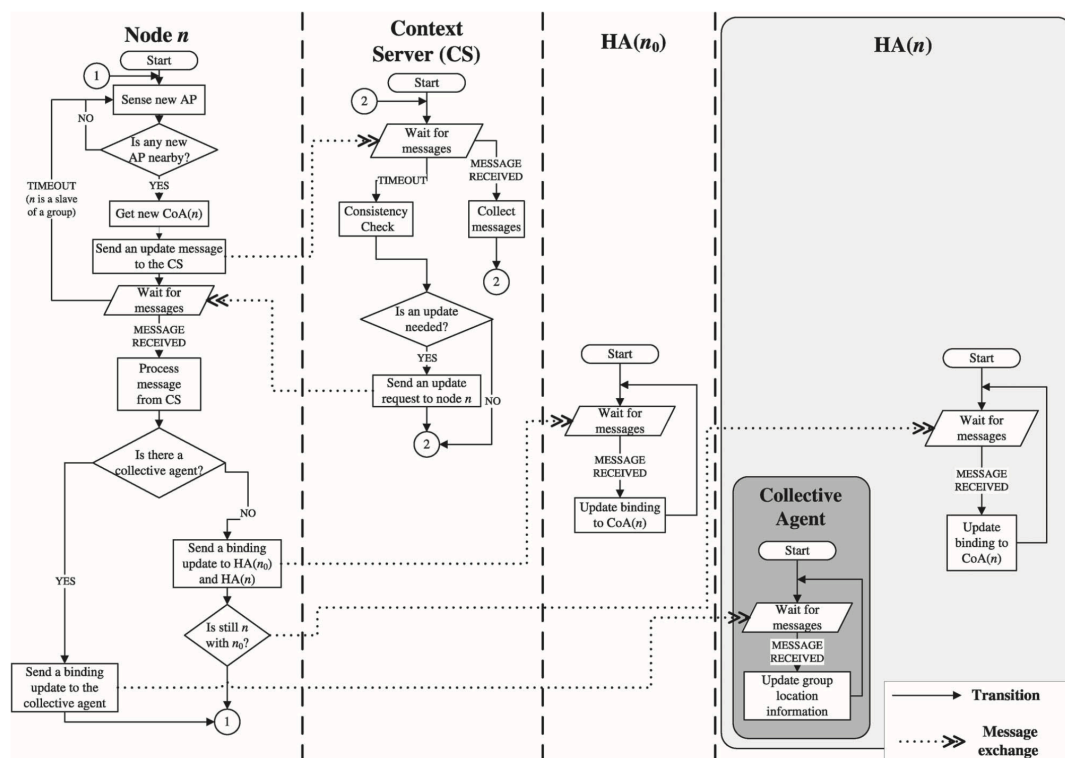
تمامی گره‌های پیروی دیگر نبوده و فقط کفایت اطلاعات موجود در HN گره سرگروه به روزرسانی شود. بدیهی است که همین امر سبب کاهش داده‌های انتقالی و در نتیجه کاهش مصرف انرژی در سیستم‌های مبتنی بر اینترنت اشیا می‌گردد.

پس از آنکه جابجایی یک گره در شبکه به وقوع پیوست، گره مذکور اطلاعات خود را از قبیل IPv6, MAC Address، فناوری ارتباطی ($\psi(n)$) و در صورتی که گره سرگروه باشد آدرسی که باید برای گره‌های پیرو Set شود را به Content Server ارسال می‌نماید. Content Server دو وظیفه اصلی بعد از دریافت این اطلاعات از جانب یک گره دارد.

- به‌روزرسانی اطلاعات موقعیت و گروه‌ها: در این مرحله، CS بعد از بررسی سازگاری گره اضافه شده به گروه، به‌روزرسانی اطلاعات را انجام می‌دهد. علاوه بر این در همین حین، CS بررسی می‌کند که آیا امکان ادغام گروه موردنظر با سایر گروه‌ها وجود دارد یا خیر. به طور کلی، گروهی که گره n در آن قرار دارد ($\Gamma(n)$) اگر یکی از دو شرط مطرح شده را داشته باشد می‌تواند با گروه Γ_i ادغام گردد. شرایط بدین شرح است که یا تمام گره‌هایی با فناوری ارتباطی مشابه، اخیراً گروه Γ_i را ترک و به گروه $\Gamma(n)$ پیوسته باشند. و یا این که در گروه Γ_i دیگر گره‌ای با فناوری ارتباطی مشابه وجود نداشته باشد.

- تخمین موقعیت جغرافیایی حسگرها: بر اساس اطلاعات موقعیتی که به CS ارسال شده است، CS می‌تواند تخمینی در ارتباط با موقعیت جغرافیایی این حسگر داشته باشد. برای مثال اگر گره n به گروه جدیدی بپیوندد؛ به ازای هر گره موجود در گروه جدید (z) می‌توان رابطه $\Phi(z) = C(AP(z))$ را معرفی کرد که بیانگر زیرمحیط‌هایی است که گره z آن‌ها را پوشش می‌دهد. در نتیجه بدیهی است که این حسگرها در محیط اشتراکی به دست آمده، یعنی $\bigcap_{z \in \Gamma(n)} \Phi(z)$ قرار دارند.

در شکل ۱۶.۳ می‌توانید نمودار فرآیند جابجایی حسگرها در راهکار ارائه شده توسط این مقاله را مشاهده نمایید.



شکل ۱۶.۳: فرآیند جابجایی حسگرها در OGM

۴.۳ راهکار (SoT) Self Organized Things

در راهکار ارائه شده‌ی مقاله‌ی [۱۰]، علاوه بر دو هدف اصلی که تا به حال درباره‌ی آن صحبت نموده ایم - یعنی کاهش انرژی مصرفی سیستم و به دنبال آن، افزایش طول عمر سیستم - هدف دیگری نیز مدنظر قرار گرفته است و آن کاهش نیازمندی به انسان جهت مدیریت سیستم‌های مبتنی بر اینترنت اشیا می‌باشد. راهکار ارائه شده در این مقاله، بر اساس سه تابع Self-Configuration، Self-Optimization و Self-Healing رفتار می‌کند که در ادامه به بررسی هریک از آن‌ها می‌پردازیم.

۱.۴.۳ تابع Self-Configuration

اولین تابع به کار رفته در سازوکار ارائه شده، تابع Self-Configuration است. این تابع همان گونه که در شکل ۱۷.۳ قابل مشاهده است، وظیفه‌ی پیکربندی اولیه سیستم را بر عهده دارد. عملکرد تابع مذکور به این شکل است که هر یک از حسگرهای Trigger-Based را در حالت فعال و تمامی حسگرهای Periodic را در حالت خواب

Algorithm 1. Self-Configuration Algorithm

Require: (x, y)
Ensure: ξ_i, I_k
1: **for** $n \leftarrow 1$ to N **do**
2: **if** TBD **then**
3: $\phi \leftarrow 1$
4: **else**
5: Send Life Signal
6: $\phi \leftarrow 0$
7: **end if**
8: **end for**
9: Calculate ξ
10: Calculate I_K

شکل ۱۷.۳: شبه کد تابع Self-Configuration

قرار می‌دهد. لازم به ذکر است که در این سیستم، حالت هر حسگر (که با ϕ نمایش داده می‌شود) بر سه نوع است. خواب^۱، بیدار^۲ و منفعل (خاموش)^۳. در حالت خواب، حسگر روشن است اما به پایش و دریافت اطلاعات محیط مشغول نیست و در نتیجه کمترین میزان مصرف انرژی را دارد. در حالت بیدار (فعال)، حسگر روشن است و به پایش محیط می‌پردازد. در حالت خاموش نیز حسگر به طور کلی خاموش است و انرژی‌ای مصرف نمی‌کند. پس از آن که وضعیت هر حسگر بنابر نوع آن تنظیم شد؛ همان‌طور که در خط ۹ و ۱۰ شبه کد شکل ۱۷.۳ مشاهده می‌شود؛ نوبت به محاسبه‌ی دو پارامتر تعریف شده در این مقاله به ازای هر یک از حسگرها می‌رسد. در ادامه به تعریف این دو پارامتر می‌پردازیم.

محاسبه هم‌پوشانی (ξ_i)

یکی از پارامترهای معرفی شده در مقاله‌ی مورد بررسی، پارامتری تحت عنوان هم‌پوشانی است که مشخص‌کننده‌ی این است که یک حسگر چه مساحتی را منحصرًا تحت پوشش قرار داده است. نحوه‌ی محاسبه‌ی این پارامتر در فرمول ۱۶.۳ مشاهده می‌شود. (D_{ij} بیانگر فاصله‌ی بین دو گره i و j است).

^۱Sleep

^۲Active

^۳Passive

$$\zeta_i = \sum_j \frac{D_{ij} \times 1(D_{ij} < 2R) + 2R \times 1(D_{ij} \geq 2R)}{2R} \quad (۱۶.۳)$$

محاسبه همبستگی فضایی (I_k)

پارامتر دیگری که در این مقاله به آن اشاره شده، پارامتر همبستگی فضایی^۱ می باشد. دلیل معرفی این پارامتر این است که پارامتر قبلی (یعنی هم پوشانی ζ_i) به تنهایی به موقعیت مکانی حسگرها وابسته است و توجهی به وضعیت حسگرها ندارد.

^۱ Spatial Correlation

مراجع

[1] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, and L. Si, "Expertise retrieval," *Foundations and Trends® in Information Retrieval*, vol.6, no.2-3, pp.127-256, 2012.

[۲] مصطفی واحدی، "درختان پوشای کمینه دورنگی مسطح"، مجله فارسی نمونه، جلد ۱، صفحات ۲۲-۳۰، آبان ۱۳۸۷.