فصلنامه علمی-ترویجی پدافند غیرعامل

سال سوم، شماره ۳، پاییز ۱۳۹۱، (پیاپی ۱۱): صص ۱۹-۲۹

بررسی سیستم عامل‌های شبکه حسگر

احمد ناصری[۱]، مجید غیوری ثالث[۲]، مهدی نقوی[۳]

تاریخ دریافت: ۹۱/۰۸/۲۵
تاریخ پذیرش: ۹۱/۱۰/۰۴

چکیده

یکی از اصول پدافند غیرعامل، تشخیص به‌موقع نفوذ عامل‌های دشمن به حیطه سرزمینی یک کشور است. حسگرها به‌عنوان یکی از ابزارهای لازم برای شناسایی و تشخیص نفوذ و تهاجم دشمن مورد استفاده قرار می‌گیرند. حسگرها برای شناسایی و تشخیص و انجام وظایف در مراکز نظامی، مناطق حساس و کاربردهای دفاعی نیاز به سیستم‌عامل دارند، طراحی سیستم‌عامل شبکه حسگر با طراحی سیستم‌عامل‌های معمولی متفاوت است. در طراحی سیستم‌عامل شبکه‌های حسگر بی‌سیم باید محدودیت‌های گره‌های حسگر، مانند اندازهٔ فیزیکی، منبع انرژی، قدرت پردازش و ظرفیت حافظه را در نظر گرفت. معماری، مدل‌اجرایی، برنامه‌ریزی مجدد، زمان‌بندی و مدیریت انرژی، ویژگی‌های مهم سیستم‌عامل هستند که در انتخاب سیستم‌عامل برای شبکه حسگر در کاربردهای نظامی مفید می‌باشند. در این مقاله ما ویژگی‌های سیستم‌عامل شبکه حسگر را ارائه می‌نماییم، هدف ما از ارائه ویژگی سیستم‌عامل شبکه حسگر، انتخاب بهترین سیستم‌عامل برای حسگرهای مراکز نظامی می‌باشد. سپس تعدادی از سیستم‌عامل‌های شبکه حسگر مانند تینی، مانتیس، کانتیکی، اِس‌اُاِس ، برتا، کُرموس، نانوآرکی و آیس/پیروس را براساس این ویژگی‌ها بررسی، دسته‌بندی و مقایسه نموده و در پایان، سیستم‌عامل نانوآرکی را با توجه به خصوصیات فعالیت‌های حساس نظامی به‌عنوان یک گزینه برای حسگرها پیشنهاد می‌نماییم.

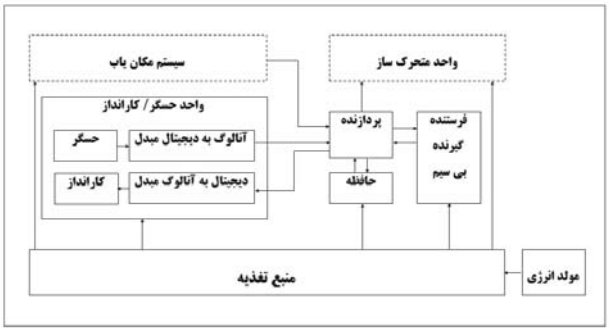
کلیدواژه‌ها: شبکه حسگر بی‌سیم، سیستم‌عامل شبکه حسگر بی‌سیم

۱- دانشجوی کارشناسی ارشد نرم‌افزار دانشگاه جامع امام حسین(ع) nasseri1355@yahoo.com- نویسنده مسئول
۲ – استادیار و عضو هیئت علمی گروه کامپیوتر دانشگاه جامع امام حسین(ع)
۳- دانش‌آموخته و عضو گروه کامپیوتر دانشگاه جامع امام حسین(ع)

۱- مقدمه

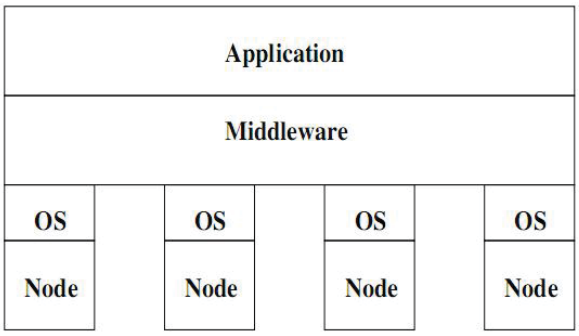
شبکه‌های حسگر بی‌سیم[1] نوعی از شبکه‌ها هستند که به‌طور معمول، از تعداد زیادی گره ارزان‌قیمت تشکیل شده‌اند، تاریخچه شبکه‌های حسگر بی‌سیم به دوران جنگ سرد و ایده اولیه آن به طراحان نظامی صنایع دفاع آمریکا برمی‌گردد. اولین شبکه بی‌سیم در سال ۱۹۷۰ توسط دارپا[2] و به دلایل نظامی به‌وجود آمدند. شبکه حسگر بی‌سیم در کاربردهای مختلفی مانند صنعت، زیست‌محیطی، پزشکی، نظامی و... مورد استفاده قرار می‌گیرد. از مهم‌ترین کاربردهای نظامی این شبکه‌ها، شناسایی و بررسی آماری تجهیزات نیروی دشمن، نحوه آرایش، مسیر حرکت نیروهای دشمن یا نیروهای خودی، وضعیت نیروهای خودی در قبال نیروهای دشمن، کنترل مرزهای آبی و خاکی، شناسایی مناطق آلوده (تشخیص آلودگی‌های شیمیایی، میکروبی، هسته‌ای) و... می‌باشد.

یک گره حسگر بی‌سیم دارای قابلت‌های ارتباطی، محاسباتی، حساسیت و ذخیره‌سازی است. این گره‌های کوچک دارای محدودیت‌هایی از نظر منابع موجود مانند قدرت پردازش، توان باطری، حافظه و پهنای باند [۱-۳] هستند. شکل (۱) ساختار گره حسگر را نشان می‌دهد. معمولا هر گره متشکل از یک میکروکنترلر، منبع تغذیه، فرستنده و گیرنده امواج رادیویی یا RF و حافظه خارجی است.



شکل ۱- معماری سخت‌افزار گره شبکه‌های حسگر [۵،۴،۱]

کارکرد اصلی یک سیستم‌عامل، پنهان‌کردن جزئیات سطح پایین گره حسگرها با ایجاد یک رابط صریح با دنیای بیرونی است. مدیریت پردازشگر، مدیریت حافظه، مدیریت تجهیزات، سیاست‌های زمان‌بندی و چندنخی و چندبرنامه‌ای، از جمله خدمات یا سرویس‌های سطح پایین هستند که باید توسط یک سیستم‌عامل تامین شود. علاوه بر این‌ها، سیستم‌عامل باید خدماتی مانند پشتیبانی برای بارگذاری پویا و باردهی قطعه‌ها، تأمین مکانیزم‌های همزمانی صحیح، واسط برنامه‌ریزی کاربردی[3] - که من‌بعد در این مقاله به اختصار واسط نام خواهیم برد - برای دسترسی به سخت‌افزارها و تقویت خط‌مشی‌های مدیریت صحیح انرژی را ارئه دهد. اگرچه تعدادی از این خدمات با خدمات سیستم‌عامل‌های معمولی یکسان هستند، اما تحقق این خدمات در شبکه حسگر بی‌سیم که دارای محدودیت‌هایی در منابع یا امکانات می‌باشد، ضروری است[۴]. شکل (۲) جایگاه سیستم‌عامل را در نرم‌افزارهای شبکه حسگر بی‌سیم نشان می‌دهد، میان‌افزارها به‌صورت توزیع‌شده روی سیستم‌عامل گره‌ها قرار دارند. هسته اصلی این سیستم‌عامل در هر گره قرار می‌گیرد. در بالای آن، میان‌افرارها به صورت قطعه‌های در تعامل با گره‌ها اجرا می شوند.



شکل ۲- لایه‌های نرم افزاری گره شبکه‌های حسگر [۵،۴،۱]

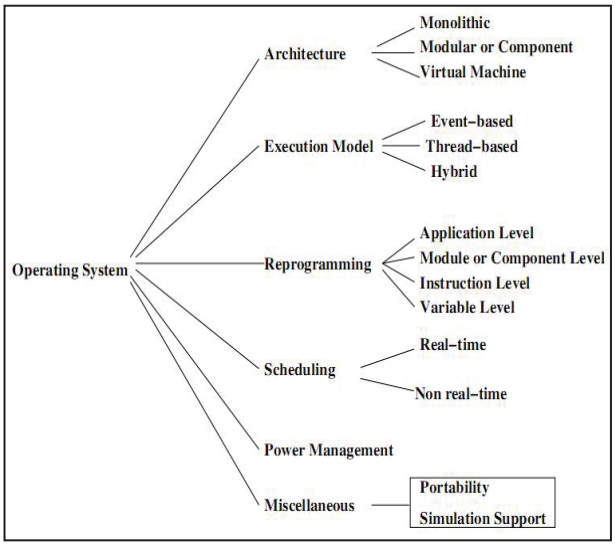
در این مقاله، ما ویژگی‌های سیستم‌عامل شبکه حسگر را مطالعه می‌کنیم که بر طراحی و انتخاب سیستم‌عامل شبکه حسگر بی‌سیم اثر می‌گذارند. ویژگی‌های اصلی سیستم‌عامل [۶] عبارت‌اند از: معماری، برنامه‌ریزی مجدد، مدل اجرا و زمان‌بندی. ویژگی‌های دیگر مانند مدیریت انرژی، پشتیبانی از شبیه‌سازی و پرتابل بودن یا قابلیت حمل نیز می‌باشد. این ویژگی‌ها که در شکل (۳) دسته‌بندی شده برای مقایسه سیستم‌عامل‌های شبکه حسگر مورد استفاده قرار می‌گیرند.

در پادگان‌ها، میدان جنگ، موشک، هواپیما، شناسایی مناطق آلوده، شناسایی و کنترل مناطق نظامی، هسته‌ای، شیمیایی و... چه سیستم‌عاملی برای گره‌های حسگر استفاده نماییم؟ مراکز نظامی دارای خصوصیات خاصی هستند که انتخاب سیستم‌عامل به این خصوصیت‌ها ارتباط دارد. ما با بیان ویژگی سیستم‌عامل‌های شبکه حسگر و مقایسه آن با خصوصیات فعالیت‌های نظامی، بهترین سیستم‌عامل را می‌توانیم پیشنهاد دهیم. سیستم‌عامل نانوآرکی یک سیستم‌عامل پیشنهادی برای گره‌های مناطق نظامی می‌باشد. این سیستم‌عامل می‌تواند گره‌های حسگر را در مراکز نظامی با توجه به خصوصیات مناطق نظامی هدایت نماید،

1- Wireless Sensor Network (WSN)

۲- پروژه آژانس تحقیقات پیشرفته دفاعی آمریکا (DARPA)

3- Application Programming Interface



شکل ۳- ویژگی‌های سیستم‌عامل شبکه حسگر[۴]

سعی ما در این مقاله پاسخ‌گویی به سؤال زیر می‌باشد:

- ✓ کدام سیستم‌عامل برای حسگرهای مناطق نظامی مفید می‌باشد؟

۲- ویژگی‌های سیستم‌عامل شبکه حسگر بی‌سیم

حوزه فعالیت سیستم‌عامل شبکه حسگر بی‌سیم در دو سطح است؛ سطح اول، شبکه و سطح دیگر گره است. ویژگی سیستم‌عامل باید در دو سطح جوابگو باشد. موضوعات مناسب در سطح شبکه عبارت‌اند از: اتصال، مسیریابی، مشخصات کانال ارتباطی، پروتکل‌ها و غیره. در سطح گره، سخت‌افزار، رادیو (بی‌سیم)، پردازشگر، حسگرها و انرژی محدود می‌باشند. ویژگی‌های مهم[۶] مربوط به سطح گره مدیریت منابع محدود، کنترل همزمان، مدیریت توان و مدیریت حافظه هستند که در آن، موضوعات مربوط به ارتباطات بین دو گره، کنترل خرابی، همگن‌بودن و مقیاس‌پذیری نیز اضافه می‌شود.

در زیر، ویژگی‌های مهم سیستم‌عامل در سطح شبکه و گره بررسی می‌شوند که باید در هنگام طراحی یک سیستم‌عامل برای شبکه حسگر بی‌سیم مدنظر قرار گیرند.

۲-۱- معماری انعطاف‌پذیر

در معماری سیستم‌عامل باید تکنیک‌هایی مانند سیستم یکپارچه، سیستم لایه‌ای و سیستم مجازی را مورد بررسی قرار دهیم. در سیستم یکپارچه، سیستم‌عامل به‌صورت یک مجموعه از رویه‌ها نوشته شده است که هر یک از آنها می‌توانند دیگری را به هنگام نیاز فراخوانی کنند. برای مخفی کردن اطلاعات امکاناتی وجود ندارد و هر رویه برای دیگر رویه‌ها کاملا قابل مشاهده است. در سیستم لایه‌ای، سیستم‌عامل به تعدادی سطح یا لایه تقسیم می‌شود که هر کدام در بالای لایه پائین‌تر قرار می‌گیرند. مزیت مهم این روش، پیمانه‌ای[۱] بودن آن است. یعنی لایه‌ها به گونه‌ای تقسیم‌بندی می‌شوند که هر لایه فقط توابع و سرویس‌های لایه پائین‌تر را استفاده می‌کند. بدین ترتیب هر لایه را می‌توان مستقل از لایه‌های دیگر طراحی کرد، بسط داد و خطایابی کرد. در معماری ماشین مجازی، ماشین‌های مجازی، مشابه یک نسخه از سخت‌افزار عریان هستند که دارای دستورات کاربر و هسته، I/O، وقفه‌ها و چیزهای دیگر (ماشین حقیقی) می‌باشند.

معماری هسته در روش ارائه خدمات اثر می‌گذارد. دو مورد که تحت تاثیر معماری سیستم‌عامل قرار دارند عبارت‌اند از: ۱- پیکربندی مجدد زمان اجرای سرویس‌ها ۲- اندازه هسته اصلی.

امکان اضافه نمودن یا به‌روزرسانی خدمات (سرویس) هسته، به معماری سیستم‌عامل بستگی دارد. اندازه هسته اصلی به معماری بستگی دارد. اگر در یک معماری امکان دسته‌بندی کردن تمام خدمات مورد نیاز در یک تصویر سیستم منحصربه‌فرد فراهم باشد، در آن صورت اندازه هسته اصلی افزایش می‌یابد. باید در نظر داشته باشیم که تمام خدماتی (سرویس‌هایی)که هسته اصلی را می‌سازند در یک زمان در اجرای برنامه‌ها مورد نیاز نیستند. در یک معماری دیگر، اگر هسته سیستم از نیازهای کاربری خاص پشتیبانی کند این کار باعث کاهش اندازه هسته می‌شود؛ اما این کار امکان اجرای چند برنامه‌گی را نمی‌دهد. از این گذشته در چنین معماری با تغییر در هسته برنامه کاربردی، باید کل تصویر هسته را جایگزین هسته قبلی نمود [۴].

اگر در معماری امکان ترکیب خدمات در زمان اجرا فراهم شود، این کار سبب کاهش اندازه هسته می‌شود. ایجاد انعطاف‌پذیری در به‌روزنمودن یا جایگزینی سرویس مربوطه بسیار اهمیت دارد و می‌تواند بدون جایگزینی تصویر کلی هسته، اصلاح یا عوض شود.

۲-۲- مدل اجرای کارآمد

برای اجرای کارآمد، روش‌های مختلفی وجود دارد ، روش چندنخی[۲] یک فرایند که برنامه‌ای در حال اجراست، می‌تواند به بخش‌ها یا نخ‌هایی (بندهایی) تقسیم شود که می‌توانند به صورت همزمان اجرا شوند. برنامه‌هایی که چند وظیفه مستقل از هم را انجام می‌دهند می‌توانند به‌صورت چندنخی نوشته شوند. روش رویدادمحور، کاربران همزمان دستوراتی را وارد کرده و پاسخ آن را سریعاً دریافت می‌کنند، هدف اصلی این سیستم‌ها پاسخ‌دهی سریع به رویدادها است.

مدل اجرا سبب خلاصه‌سازی واحد محاسباتی می‌شود و خدماتی مانند همزمان‌سازی، ارتباطات و زمان‌بندی را فراهم می‌کند. این خلاصه‌سازی‌ها توسط برنامه‌ریز برای ارائه برنامه‌های کاربردی مورد

1- Modularity
2- Multithreading

استفاده قرار می‌گیرد. سرویس ارتباطی، روش واحدهای محاسباتی را تعریف می‌کند. آنها کار تبادل داده‌ها را فراهم می‌کنند. برقراری ارتباط داده‌ها می‌تواند به اشتراک گذاشته شود. دسترسی به داده‌های به اشتراک گذاشته شده نیازمند مکانیزم‌های همزمان‌سازی صحیح است تا از شرایط وضعیت رقابتی جلوگیری شود. گاهی باید برنامه کاربردی، کارهای همزمان و سنگین را انجام دهد. سوئچینگ میان کارها برای جلوگیری از عدم اجرای وظایف ضرورت دارد. واحد محاسباتی انعطاف‌پذیر به داشتن معماری انعطاف‌پذیر برای سیستم کمک می‌کند. زمان‌بندی واحدهای محاسباتی در کاربردهای اساسی ماموریتی بسیار مهم است زیرا اجرای آنها بعد از ضرب‌الاجل منجر به مشکلات جدی می‌شود[۴].

برنامه کاربردی می‌تواند گاهی چند کار را انجام دهد. این‌کار نیازمند زمان‌بندی صحیح پردازشگر برای اجرای این کارها است. زمان‌بندی، ترتیب اجرای برنامه‌ها توسط پردازشگر را مشخص می‌کند.

۲-۳- واسط برنامه‌ریزی کاربردی صریح

واسط‌های برنامه‌ریزی کاربردی نقش مهمی در جداسازی کارهای سطح پایین گره و برنامه کاربردی دارند. سیستم‌عامل باید مجموعه جامعی از واسط‌ها را ارائه دهد که با سیستم و I/O آن تعامل داشته باشند. این کار کمک می‌کند که ارائه برنامه‌ها بدون توجه به کارکرد سطح پایین سخت‌افزار گره حسگر، انعطاف داشته باشند. واسط‌های برنامه‌ریزی کاربردی شامل بخش‌های زیر است[۴]:

- واسط شبکه (ارسال و دریافت فعالیت‌ها)
- واسط خواندن داده‌های حسگر
- واسط‌های کنترل حافظه (فعالیت‌های بارگذاری و ذخیره‌سازی)
- واسط مدیریت توان (خواب، خواندن سطح انرژی)
- واسط‌های مدیریت وظیفه (تعیین تاخیرها، تعیین اولویت‌ها و وظایف‌ها)

این واسط‌ها به ارائه‌دهنده برنامه امکان می‌دهند که برنامه‌های کاربردی را ایجاد کند و از امکانات موجود استفاده مؤثر بنماید. واسط‌ها مربوط به دسترسی به حافظه برای پیکربندی مجدد نرم‌افزارهای مهم هستند که به‌صورت پویایی در گره حسگر اجرا می‌شوند واسط‌های تعیین تاخیرهای کارها سبب انعطاف‌پذیری برنامه‌نویس در زمان‌بندی فعالیت‌ها می‌شود.

۲-۴- برنامه‌ریزی مجدد

برنامه‌ریزی مجدد یک ویژگی الزامی برای سیستم‌عامل است و سبب تسهیل مدیریت نرم‌افزار در گره‌های حسگر می‌شود. این روش، به‌روزرسانی پویای نرم‌افزار است که در گره‌های حسگر اجرا می‌شود. برنامه‌ریزی مجدد در شبکه حسگر بی‌سیم بسیار مورد توجه است؛ زیرا بعد از استفاده و به خاطر وجود تعداد زیادی گره‌های حسگر در شبکه، امکان دسترسی به آنها وجود ندارد. بدون برنامه‌ریزی مجدد، اضافه‌کردن، اصلاح یا حذف نرم‌افزار از سیستم در حال اجرا در شبکه حسگر بی‌سیم دشوار است[۴].

برای به‌روزرسانی نرم‌افزار در گره‌ها، کد با استفاده از پروتکل‌های انتشار، در فضا [۷, ۸]توزیع می‌شود. این پروتکل‌ها برای تقسیم و ترکیب کدها می‌باشند که باید برای به‌روزرسانی نرم‌افزار در گره‌ها ارسال شوند. ارتباطات در این پروتکل‌ها تک‌گام[1] و چندگام[2] است. در روش یک‌گام، گره‌ها مستقیما از طریق سیم یا بی‌سیم به ایستگاه پایه وصل هستند و سپس مجددا برنامه‌ریزی می‌شود. در روش ارتباط چندگام این کد به صورت گره به گره در شبکه ارسال می‌شود و بعد از دریافت کد توسط گره، نرم‌افزارش به‌روزرسانی می‌شود. این کار نیازمند یک روش مدیریت موثر است.

برای موفقیت برنامه‌ریزی مجدد در هر زمان در هنگام اجرای سیستم، کد گره باید قابل جاگذاری مجدد باشد. کد قابل جاگذاری مجدد، مستقل از جایگاه است و می‌تواند از هر مکانی از حافظه اجرا شود. این یک نیاز مهم برای برنامه‌ریزی مجدد است؛ چون کد اصلاح شده باید در هر بخش از حافظه آزاد موجود، بارگذاری یا اجرا شود.

۲-۵- زمان‌بندی

زمان‌بندی واحدی است که ترتیب اجرای فعالیت‌ها توسط پردازنده را مشخص می‌نماید. هدف از زمان‌بندی، کاهش بیکاری پردازنده و اجرای سریع‌تر فعالیت‌ها است. در سیستم‌عامل، الگوریتم‌های مختلفی برای زمان‌بندی مانند FIFO، اولویت، نوبت چرخشی، FCFS، بالاترین نسبت پاسخ، کوتاه‌ترین کار، بخت‌آزمایی تضمین شده و... وجود دارد. انتخاب الگوریتم زمان‌بندی بستگی به کاربرد و محیط دارد.

سیستم‌عامل براساس زمان، به دو دسته بی‌درنگ[3] و غیربی‌درنگ تقسیم می‌شود. برنامه‌های کاربردی بی‌درنگ را می‌توان به دو دسته متناوب و غیرمتناوب یا بحرانی و غیربحرانی تقسیم کرد. نمونه کلاسیک متناوب، برنامه دیدبانی است که در آن، داده‌ها از محیط خوانده و با شیوه متناوب پردازش می‌شوند. هدف‌یابی، انفجار و حریق، نمونه‌ای از وظایف غیرمتناوب هستند. این مثال‌ها را می‌توان با عنوان بحرانی و غیربحرانی نیز طبقه‌بندی کرد.

سیستم‌عامل‌های بی‌درنگ در کاربردهای بحرانی به‌کارگرفته می‌شوند. به‌عنوان مثال، در کاربردهایی مانند کشف حریق در راکتورهای هسته‌ای، اقدامات بازدارنده‌ای در دیدن حریق دقیقاً انجام می‌شوند. محدودیت‌های سیستم‌عامل‌های بی‌درنگ برنامه‌ها را به کمک یک بخش زمان‌بندی بی‌درنگ می‌توان به نتیجه رساند[۹].

1- Single-hop
2- Multi-hop
3- Real-time

۲-۶- مدیریت منابع

یکی از وظایف اصلی یک سیستم‌عامل، مدیریت مؤثر منابع سیستم است. منابع موجود در یک گره حسگر معمولی: پردازشگر، حافظه، برنامه، باتری، حسگرها و... هستند. استفاده مؤثر از پردازشگر، مستلزم استفاده از یک سیاست زمان‌بندی بهینه است. استفاده از حافظه، مستلزم حفاظت از حافظه، تخصیص حافظه پویا و... است. باتری باید به‌عنوان یک منبع خاص مورد توجه باشد. مدهای خواب به مدیریت توان باتری کمک می‌کنند. مدیریت حسگرها شامل کنترل نرخ دریافت است. مسئولیت سیستم‌عامل استفاده از مکانیزم‌های ضروری موجود است تا توان را به صورت بهینه مصرف کند [۴].

۳- بررسی سیستم‌عامل‌های شبکه حسگر براساس ویژگی‌های سیستم‌عامل

در قسمت قبل، ما ویژگی‌های مهم طراحی را بیان نمودیم و در ادامه، سیستم‌عامل‌های شبکه حسگر را براساس این ویژگی‌ها بررسی و دسته‌بندی می‌نماییم.

۳-۱- معماری

یکی از ویژگی‌های سیستم‌عامل، معماری می‌باشد که در روش ارائه خدمات اثر می‌گذارد. ما در زیر، تعدادی از سیستم‌عامل‌های شبکه حسگر را براساس ویژگی معماری بررسی می‌کنیم.

سیستم‌عامل‌هایی که در گروه یکپارچه قرار می‌گیرند، تینی[1] ، MagnetOS و نانوآرکی[2] هستند[۱۰]؛ یعنی از مدل جزء در زمان کامپایل و یک تصویر ایستای منفرد در زمان اجرا استفاده می‌کنند. سیستم‌عامل‌های مانتینس[3] [۱۱]، کانتیکی [۱۲]، اِس اُ اِس[4]، برتا[5] [۱۳] و سیستم‌عامل کُرموس[6] از روش پیمانه‌ای استفاده می‌کنند. سیستم‌عامل کانتیکی[7] سرویس‌های را ارائه می‌کند که باید در فرآیندها استفاده شود، اِس‌اُاِس [۱۴] از روش پیمانه‌ایی در هسته و در سطح برنامه کاربردی استفاده می‌کند. این کار در اِس اُ اِس با استفاده از مجموعه‌ای از تعامل قطعه‌ها انجام می‌شود.

۳-۲- مدل اجرایی کارآمد

مدل اجرایی، یکی از ویژگی‌های سیستم‌عامل است که سبب خلاصه‌سازی واحد محاسباتی می‌شود و خدماتی مانند همزمان‌سازی، ارتباطات و زمان‌بندی را فراهم می‌کند. مدل اجرایی به چند دسته تقسیم می‌شود که در ادامه، سیستم‌عامل‌های شبکه حسگر را براساس این ویژگی‌ها مورد بررسی قرار می‌دهیم.

1- TinyOS
2- Nano-RK
3- MantisOS
4- SOS
5- Bertha
6- CORMOS
7- Contiki

۳-۲-۱- مدل اجرایی رویدادمحور

رویدادمحور، یکی از روش‌های پیاده‌سازی فرایندها در سیستم‌عامل می‌باشد. در ادامه، تعدادی از سیستم‌عامل‌های شبکه حسگر را که از این ویژگی‌در پیاده‌سازی فرایندها استفاده می‌کنند مورد بررسی قرار می‌دهیم

- **سیستم‌عامل تینی:** این سیستم‌عامل، یک سیستم‌عامل رویدادمحور[۱۵] است که یک قالب[8] برنامه‌نویسی را برای سیستم‌های تعبیه‌شده[9] ارائه می‌دهد. این سیستم‌عامل دارای یک مدل اجرایی مبتنی بر اجزاء است که در nesC اجرا می‌شود و دارای یک حافظه بسیار کم است. مدل همزمانی تینی براساس دستورات، رویدادهای غیرهمزمان، محاسبات موسوم به وظایف و واسطه‌های فاز جداسازی است. کاربر برنامه باید کنترل‌کننده‌هایی را بنویسد که به راه‌اندازی یک رویداد کمک کند. کنترل‌کننده‌ها، دستورات و رویدادها می‌توانند با زمان‌بندی ترتیبی اجرا شوند. وظایف تا تکمیل کار، اجرا می‌شوند. شرایط رقابتی ناشی از تقدم را می‌توان با بخش‌های اتمی حل کرد. معماری ارتباطی سیستم‌عامل تینی، از مفهوم پیام فعال[10] استفاده می‌کند و در آن بسته‌ها ۳۶ بایت و یک بایت راه‌انداز دارند. یک گره پس از دریافت پیام فعال، آن را به کنترل‌کننده متناظر ارسال می‌کند. مدل مبتنی بر رویداد سیستم‌عامل تینی دارای معایبی مانند عدم انعطاف‌پذیری برنامه‌نویسی و عدم تقدم است که به خاطر استفاده از مدل رویداد است[۱۶] .
- **سیستم‌عامل اِس‌اُ اِس:** این سیستم‌عامل در زبان c توسعه یافته و از مدل برنامه‌نویسی رویدادمحور تبعیت می‌کند. یک برنامه در اِس‌اُ اِس به‌صورت یک یا چند قطعه پیاده‌سازی می‌شود. قطعه‌ها باینری‌های مستقل از جایگاه هستند که یک کار را انجام می‌دهند و به سرویس‌ها در سیستم‌عامل کانتیکی و وظایف در سیستم‌عامل تینی شبیه هستند. قطعه‌ها در اِس‌اُاِس دارای مدخل ورودی و خروجی هستند. ورود به قطعه تنها از طریق یکی از روش‌های زیر است: یکی پیام‌های تحویلی از بخش زمان‌بندی و دیگری فراخوانی به بخش‌های ثبت‌شده با قطعه‌ها برای کاربرد بیرونی است. هر قطعه دارای یک بخش کنترل‌کننده برای کنترل پیام است. بخش‌های لینک پویا در هسته، امکان بارگذاری قطعه‌های باینری پویا را فراهم می‌کنند. اِس‌اُ اِس از صف‌های اولویت برای پیام زمان‌بندی استفاده می‌کند که برخلاف صف ترتیبی در سیستم‌عامل تینی است.
- **سیستم‌عامل آیس/پیروس:** سیستم‌عامل آیس[11] با هدف رسیدن به اهدافی مانند اندازه کوچک، اطلاع از توان، توزیع و پیکربندی

8- Frame Work
9- - Embedded
10- Active Messages(AM)
11- EYES

مجدد آغاز شد. سیستم‌عامل آیس یک مدل اجرایی رویدادمحور را اتخاذ نموده تا به هدف اندازه کوچک کد و انرژی محدود موجود دست یابد. موجودیت محاسباتی مبنا، وظیفه‌ای است که بخشی از یک کد است که تا تکمیل اجرا می‌شود. استفاده موثر در منابع توزیع‌شده در شبکه با مدیریت منابع و مکانیزم‌های فراخوانی راه دور کنترل می‌شود و دو لایه انتزاعی برای برنامه‌نویسی فراهم می‌کند. هر کدام از این لایه‌ها مجموعه‌ای از واسط‌ها را برای توسعه‌دهنده فراهم می‌کند. سطح اول، موسوم به سطح واسط لایه شبکه حسگر است که واسط‌های مرتبط با خواندن داده حسگر، اطلاعات مربوط به منابع را فراهم می‌کند. همچنین واسط‌های مربوط به داده‌ها، انتقال داده‌ها و جمع‌آوری اطلاعات شبکه را فراهم می‌نماید. لایه دوم (واسط لایه اول)، لایه سیستم‌های توزیع‌شده است که دسترسی شفاف به گره‌های حسگر از راه دور را فراهم می‌کند[۱۷].

سیستم‌عامل آیس/پیروس ویژگی‌های سیستم‌عامل آیس را با زمان‌بندی بی‌درنگ، مدیریت حافظه، و مدیریت منابع سیستم‌عامل پیروس[1] گسترش داده‌اند. سیستم‌عامل پیروس، گرانولاریتی‌های (دانه‌دانه‌بودن) بی‌درنگ برای برنامه‌های کاربردی را فراهم می‌کند. پیروس دارای وظیفه انتزاع کاری است. هر وظیفه دارای اولویتی مرتبط با آن است. پیروس چندبرنامه‌گی مبتنی بر اولویت را برای پاسخ سریع به رویدادها ارائه می‌دهد. چندبرنامه‌گی مبتنی بر اولویت یا رویدادمحور به کمک بخش زمان‌بندی انجام می‌شود که از الگوریتم EDFI [۱۸]تبعیت می‌کند. مدیریت ذخیره‌سازی و ارتباط به ترتیب با سیستم پیام‌دهی و مدیریت قطعه انجام می‌شود.

- **سیستم‌عامل کُرموس:** انتزاع‌های اصلی که سیستم‌عامل کُرموس [۱۹] ارائه می‌کند عبارت‌اند از: رویدادها، کنترل کننده‌ها و مسیرها. رویدادها صریحاً با کنترل‌کننده‌ها ایجاد می‌شوند. کنترل‌کننده‌ها توابعی هستند که تمام کار پردازش را در هر گره انجام می‌دهند. هر کنترل‌کننده، وقتی مورد حمایت قرار می‌گیرد که با بخش زمان‌بندی رویداد زمان‌بندی شود. سیستم‌عامل کُرموس از یک بخش زمان‌بندی نزدیک‌ترین سررسید[2] اول استفاده می‌کند که زمان‌بندی رویدادها برای برنامه‌های کاربردی را راحت می‌کند و وجود واسطه‌های غیرهمزمان مبتنی بر تایمر را از برنامه‌های کاربردی پنهان می‌کند. بخش‌های زمان‌بندی سیستم‌عامل کُرموس در زمانی که محاسباتی برای انجام نباشد پردازشگر را به وضعیت حالت خواب می‌برد. کنترل‌کننده‌ها اتمی و اجرا تا تکمیل هستند و نباید هرگز بلوکه شوند. جدول (۱) دسته‌بندی سیستم‌عامل‌های شبکه حسگر را براساس مدل اجرایی نشان می‌دهد.

1- PEEROS
2- Deadline

جدول ۱- مدل اجرای سیستم‌عامل‌های شبکه حسگر[۴]

Event-based	Thread-based	Hybrid	Others
TinyOS	MantisOS	Contiki (Event+Thread)	SenOS
SOS		kOS (Event+Object)	Nano-RK
CORMOS			
EYES			
PEEROS			

۳-۲-۲- مدل اجرایی نخ‌محور

یکی دیگر از روش‌های مدل اجرایی، نخ‌محور[3] است که استفاده از روش نخ‌محور در طراحی سیستم‌عامل باعث ایجاد چندبرنامه‌گی می‌شود. در ادامه، ما تعدادی سیستم‌عامل را براساس این ویژگی مورد بررسی قرار می‌دهیم:

- **سیستم‌عامل مانتیس:** این سیستم‌عامل، یک سیستم‌عامل نخ‌محور است. نخ، یک موجودیت محاسباتی ساده است که حالت خود را دارد. مدل نخ‌محور، انعطاف نوشتن برنامه‌های کاربردی را بیشتر می‌کند؛ زیرا توسعه‌دهنده، اندازه وظیفه را که در مدل رویدادمحور اجباری است در نظر نمی‌گیرد. اجرای یک برنامه کاربردی مستلزم اجرای چند نخ است. علاوه براین نخ‌ها، یک نخ idle نیز وجود دارد که وقتی اجرا می‌شود که تمام نخ‌ها مسدود باشند. نخ idle از زیربرنامه‌های مورد نیاز برای مدیریت توان حمایت می‌کند. برای حفظ نخ‌ها، هسته، یک جدول مشخصات نخ دارد که متشکل از اولویت نخ، اشاره‌گر به کنترل نخ و اطلاعات دیگر در مورد نخ است. زمان‌بندی میان نخ‌ها به وسیله بخش زمان‌بندی اجرا می‌شود که از الگوریتم زمان‌بندی اولویت‌محور با روش round-robin تبعیت می‌کند. شرایط رقابتی با استفاده از سمافورهای شمارشی و باینری به‌وجود نمی‌آید.

۳-۲-۳- هیبریدی یا پیوندی

یکی دیگر از روش‌های مدل اجرایی، هیبریدی می‌باشد. در این روش، از خواص رویدادمحور و نخ‌محور در طراحی سیستم‌عامل استفاده می‌شود.

- **سیستم‌عامل کانتیکی:** این سیستم‌عامل، مزایای نخ و رویدادمحور را دارد و یک مدل رویدادمحور است؛ اما از چندنخی به‌صورت یک برنامه اختیاری حمایت می‌کند. برنامه اگر نیازی به چندنخی داشته باشد می‌تواند به این کتابخانه پیوند زند. رویدادها در سیستم‌عامل کانتیکی به‌صورت همزمان و غیرهمزمان دسته‌بندی می‌شوند. رویدادهای همزمان بلافاصله زمان‌بندی می‌شوند و رویدادهای غیرهمزمان بعدا زمان‌بندی می‌شوند. مکانیزم انتخابی برای جلوگیری از شرایط رقابتی استفاده و در سیستم‌عامل کانتیکی به صورت سرویس اجرا می‌شود. هر

3- Thread-based

سرویس دارای واسط و پیاده‌سازی است. برنامه تنها با واسط ارتباط دارد. پیاده‌سازی سرویس می‌تواند در زمان اجرا عوض شود. این کار با یک کتابخانه ریشه انجام می‌شود که برای دسترسی به سرویس‌ها به این برنامه متصل است.

۳-۲-۴- سایر سیستم‌عامل‌ها

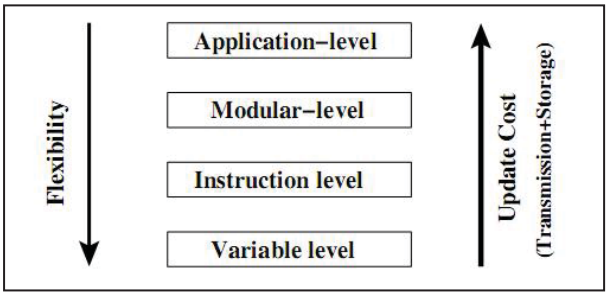
- **سیستم‌عامل سنsاُ اِس:** این سیستم‌عامل مبتنی بر ماشین حالت محدود است. اجزای اصلی سنsاُ اِس صف رویداد، تحت کتابخانه callback و یک جدول انتقالی حالت هستند. هسته سنsاُ اِس، صف رویدادها را جمع می‌کند و اقدامات مناسب و متناظر را انجام می‌دهد که سبب انتقال حالت‌ها می‌شود. کتابخانه callback مجموعه‌ای از کتابخانه‌ها را برای برنامه‌نویس فراهم می‌کند تا به سخت‌افزار دسترسی داشته باشد و رویداد و ارتباطات را مدیریت کند. هر جدول انتقال، یک برنامه را انتخاب می‌کند که در شبکه حسگر بی‌سیم اجرا می‌شود. همزمانی برنامه‌ها به کمک جداول انتقال سوئیچینگ جهت اجرا می‌باشد.
- **سیستم‌عامل نانوآرکی:** یک سیستم‌عامل بی‌درنگ مطلع از انرژی مبتنی بر صرفه‌جویی است که در شبکه حسگر بی‌سیم استفاده می‌شود. واحدهای محاسباتی و وظایف، به اولویت‌ها ارتباط دارند و وظیفه با اولویت بالاتر همواره بر وظیفه کم‌اولویت‌تر مقدم است. در وظیفه‌های حساس زمانی در برنامه‌ها، این سیستم‌عامل یک الگوریتم زمان‌بندی یکپارچه – نرخ را برای وظیفه‌ها اجرا می‌کند و به‌گونه‌ای که تأخیرهای اجزای وظایف را در نظر می‌گیرد. برنامه کاربردی می‌تواند شرایط منابع خود و تأخیرهایی که باید مورد توجه باشند را تعریف کند. همچنین انتزاع‌های سوکت را برای ارتباطات ارائه می‌کند.

تینی صرفاً از مدل رویدادمحور استفاده می‌کند، در حالی که سیستم‌عامل مانتیس از مدل نخ‌محور استفاده می‌کند. سیستم‌عامل کانتیکی از مدل هیبریدی تبعیت و از مدل رویدادمحور در سطح هسته استفاده می‌کند و نخ‌محور را مانند کتابخانه کاربردی می‌داند. سیستم‌عامل مانتیس از زیرمجموعه نخی POSIX استفاده می‌کند. اِساُ اِس هرگز جایگاه محکمی در این موضوع نداشته است زیرا بیشتر روی قابلیت پراکندگی مجدد متمرکز است. آنها از یک مدل همزمانی استفاده می‌کنند که شبیه به مدل رویداد تینی بوده و فاقد سرباری سوئیچینگ سیستم‌عامل مانتیس است.

۳-۳- برنامه‌ریزی مجدد

شکل (۴) سطوح مختلف دانه‌بندی[1] را نشان می‌دهد که سیستم‌عامل می‌تواند از برنامه‌ریزی مجدد پشتیبانی کند. این دسته‌بندی بیشتر روی گرانیولاریتی ریزتری متمرکز است که یک سیستم‌عامل پشتیبانی می‌کند. همانطور که در این شکل ملاحظه می‌شود، انعطاف‌پذیری با شرایط رفتار برنامه کاهش یافته و در آن از بالا به پایین هزینه به‌روزرسانی افزایش می‌یابد.

در ادامه، ما سیستم‌عامل‌ها را براساس سطوح شکل (۴) مورد بررسی قرار می‌دهیم.



شکل ۴- شکل تغییرات هزینه و قابلیت برنامه‌ریزی مجدد [۴].

۳-۳-۱- سطح کاربری

- **سیستم‌عامل تینی:** رفتار برنامه سیستم‌عامل تینی را می‌توان از طریق انتقال کد سخت عوض کرد و یا با اصلاح متن کد برنامه انجام داد و کد را مجددا کامپایل و تصویر جدیدی را در گره حسگر قرار داد. این برنامه از XNP به‌عنوان پروتکل انتشار کد استفاده می‌کند. این کار سبب اضافه‌بار ارتباطی بالا می‌شود؛ زیرا به‌خاطر معماری یکپارچه تینی، تصویر کل سیستم باید برای یک تغییر کوچک مجددا جایگذاری شود.

در این حالت هیچ حفاظت حافظه‌ای وجود ندارد و این خصوصیت تینی می‌باشد. هیچ مفهوم حافظه مجازی یا مدیریت حافظه در تینی وجود ندارد و کد به صورت فیزیکی متصل به آدرس است.

- **سیستم‌عامل سنsاُ اِس:** این سیستم‌عامل به صورت پویا تغییرات را به‌روز می‌نماید و آن را به جدول مخصوص برنامه‌ها انتقال می‌دهد.

۳-۳-۲- سطح ماژولار

- **سیستم‌عامل اِساُ اِس:** انگیزه و هدف اصلی توسعه سیستم‌عامل اِساُ اِس، قابلیت پیکربندی آن است که عبارت است از توان اضافه نمودن، به‌روزرسانی و حذف قطعه‌های نرم‌افزاری در زمان اجرا [۲۰]. این سیستم‌عامل از طریق سیستم‌های قطعه قابل بارگذاری (گرانیولاریتی قطعه)، از به‌روزرسانی کد اصلی پویا پشتیبانی می‌کند. قطعه‌ها را در حین اجرا می‌توان حذف و به‌روزرسانی نمود. به‌روزرسانی ضروری را در بلوک‌های توابع می‌توان انجام داد. برنامه‌ریزی مجدد در اِساُ اِس چیزی مابین سیستم‌عامل‌های تینی، Deluge وMat است؛ زیرا در سطح قطعه کار، برنامه‌ریزی را مجدد انجام می‌دهد. اِساُ اِس شامل یک مکانیزم انتشار – اشتراک است که شبیه به MOAP است و از آن

1- Granularity

در قطعه‌های توزیع‌شده در شبکه استفاده می‌گردد و براساس نیازهای کاربر می‌تواند از MOAP،Deluge و یا هر پروتکل توزیع کد دیگری استفاده کند. چون نیازی به اجرای قطعه‌ها یا تصاویر مشابه در گره‌های حسگری مجزا نداریم، می‌توان از هر دسته پروتکل‌های توزیع استفاده کرد که از جمله آنها می‌توان به پرش‌تکی[1] یا پرش چندگانه[2] اشاره کرد.

امکانات بارگذاری پویا در اِس‌اُ اِس با مشکلات زیادی در امر تخصیص حافظه و مدیریت پیام‌های قطعه‌ها مواجه است. فقدان خط مشی‌های صحیح مدیریت حافظه ممکن است سبب ایجاد مشکلاتی مانند حافظه ناکافی برای بارگذاری یک قطعه شود. مسائل و مشکلات مربوط به مدیریت پیغام عبارت از ارسال پیام به یک قطعه موجود و یا تحویل یا ارسال پیام به یک اداره‌کننده[3] نادرست است.

- **سیستم‌عامل مانتیس:** امکان برنامه‌ریزی مجدد که در سیستم‌عامل مانتیس به صورت یک کتابخانه در هسته می‌باشد، در برنامه کاربردی از این کتابخانه برای فراخوانی جهت نوشتن کد جدید استفاده می‌شود و برای اجرای کد به‌روزرسانی‌شده نصب می‌گردد. این کار نیازمند راه‌اندازی مجدد یا تنظیم مجدد نرم‌افزار است. برنامه‌ریزی پویای مجدد در سیستم‌عامل مانتیس در حال حاضر صرفا محدود به ورود از راه دور (login)، تغییر پارامترها و متغیرها است. مانتیس یک فراخوانی سیستمی را برای برنامه‌ریزی مجدد تهیه می‌کند و از برنامه‌ریزی مجدد برای کل سیستم‌عامل پشتیبانی می‌کند.
- **سیستم‌عامل کانتیکی:** در این سیستم‌عامل، اجرای سرویس را می‌توان در زمان اجرا عوض کرد. این کار توسط یک کتابخانه ریشه انجام می‌شود که به برنامه دسترسی به خدمات، لینک دارد. بارگذاری و عدم بارگذاری پویای سرویس‌ها را می‌توان به صورت انعطاف‌پذیر در سیستم‌عامل کانتیکی انجام داد. در کانتیکی به جای جاگذاری، تصویر کل سیستم همانند سیستم‌عامل مانتیس انجام می‌شود و تنها در سرویس برنامه مورد نیاز، اجازه برنامه‌ریزی مجدد را می‌دهد. سیستم‌عامل کانتیکی از روش‌های مدیریت صحیح حافظه استفاده نمی‌کند و این سبب ایجاد یک سربار در زمان برنامه‌ریزی مجدد می‌گردد. فرض بر این است که وابسته به جایگاه است. بنابراین کد باید در همان مکان حافظه بارگذاری شود. این کار در صورت افزایش اندازه کد سبب مشکلاتی در تخصیص حافظه خواهد شد.
- **سیستم‌عامل کُرموس:** هسته این سیستم‌عامل شامل برنامه‌ریز، تخصیص‌دهنده حافظه و ریجستری قطعه است، مدیر حافظه، یک جدول استاتیکی برای تخصیص و عدم تخصیص رخدادها دارد. اندازه این جدول در زمان کامپایل نمودن تعیین می‌شود. یک قطعه ممکن است هر قطعه دیگری را بارگذاری یا عدم بارگذاری نماید و نیز می‌تواند در هنگام اجرا، اداره‌کننده‌های ریجستری را بارگذاری کند، به این ترتیب، طراح این امکان را پیدا می‌کند. که یک پروتکل مسیریابی خاص یا یک پروتکل قابل اعتماد را در صورت نیاز به صورت یک قطعه فعال یا غیرفعال کند. کُرموس بر خلاف سیستم‌عامل تینی، ارتباطات را با پردازش در سیستم اجزای پایه خود تلفیق می‌کند.
- **سیستم‌عامل نانوآرکی:** یکی از اهداف نانوآرکی، تسهیل و ساده‌سازی کار پدیدآورنده‌های نرم‌افرارها بود. به این ترتیب به آنها این امکان داده می‌شد تا در یک نمونه چندوظیفه‌ای کار کنند. این، سبب ایجاد یک منحنی یادگیری کوتاه، توسعه سریع نرم‌افزارها و بهبود بهره‌وری می‌شد. از آنجا که نانوآرکی یک سیستم‌عامل چندوظیفه‌ای انحصاری است، نیازمند ذخیره‌سازی متن وظیفه جاری قبل از زمان‌بندی و برنامه‌ریزی برای وظیفه جدید است. ذخیره‌سازی وضعیت یا حالت هر وظیفه، سبب مصرف بخش زیادی از حافظه و سوئیچ‌های متعدد در متن‌ها می‌شود که کارایی را کاهش و مصرف انرژی را بالا می‌برد. در نانوآرکی هر وظیفه دارای یک سطر در بلوک کنترل وظیفه[4] است. توصیه می‌شود که جدول کنترل وظیفه، در زمان راه‌اندازی و ایجاد تصویر سیستم استفاده شود. این جدول، محتوای ریجسترها، و اولویت، زمان وقوع، اندازه نگهداری (CPU، شبکه و سنسورها) و شناسه‌های پورت فعالیت‌ها را ذخیره می‌کند. نانوآرکی براساس زمان وقوع، دو لیست مرتبط از اشاره‌گرهای TCB دارد تا دستور اجرای مجموعه‌ای از وظایف فعال و معوق را صادر نماید.

۳-۵- زمان‌بندی

سیستم‌عامل براساس زمان به دو دسته بی‌درنگ و غیربی‌درنگ تقسیم می‌شود. جدول (۲) دسته‌بندی سیستم‌عامل‌های شبکه حسگر را براساس زمان‌بندی نشان می‌دهد

جدول ۲- دسته‌بندی سیستم‌عامل‌ها براساس زمان‌بندی[۴]

Real-time	Non Real-time
Nano-RK	TinyOS
CORMOS	SOS
PEEROS	Contiki
Nano-QPlus	MantisOS
DCOS	EYES
t-kernel	SenOS
	OSSTAR
	MagnetOS
	kOS
	T2

سیستم‌عامل نانوآرکی زمان‌بندی اولویت را در دو سطح فرآیند و شبکه فراهم می‌کند. نانوآرکی برای پشتیبانی از برنامه‌های بی‌درنگ،

1- Single-hop
2- Multi-hop
3- Handling

4- Task Control Block

از الگوریتم زمان‌بندی مبتنی بر اولویت[1] استفاده می‌کند؛ یعنی در هر لحظه، وظیفه‌ای که دارای بالاترین اولویت است به‌وسیله سیستم‌عامل اجرا می‌شود. الگوریتم زمان‌بندی یکپارچه آهنگ[2] برای وظایف بی‌درنگ استفاده می‌شود و اولویت وظیفه به‌صورت استاتیکی و براساس دوره معین کار، تعیین می‌شود. هرچه زمان کار کوتاه‌تر باشد، اولویت بالاتر است.

۳-۶- مدیریت انرژی

مدیریت مصرف انرژی سیستم، شامل مدیریت مصرف انرژی واحدهای پردازشگر و فرستنده می‌باشد

- **سیستم‌عامل تینی:** این سیستم‌عامل واسطهایی را دارد که این واسطها به مصرف و مدیریت صحیح توان کمک کند. واسطها فرستنده و پردازشگر را مدیریت می‌کنند، برنامه‌ها باید یک تابع[3] را در رویه[4] فراخوانی کنند. به این ترتیب پردازشگر می‌تواند هر موقع امکان داشت به خواب برود و تا شرایط بعدی و کار بعدی در خواب باشد.
- **سیستم‌عامل مانتیس:** این سیستم‌عامل با استفاده از سیستم زمان‌بندی توان بهینه، در مصرف انرژی صرفه‌جویی می‌کند که بعد از اتمام کار نخ‌های فعال، تابع sleep() پردازشگر حسگر را خاموش می‌کند. این تابع شبیه به تابع UNIX sleep() است. ابتدا این برنامه باید تابع[5] فرمان صرفه‌جویی در مصرف انرژی را فراخوانی و فعال کند. این کار باید قبل از فراخوانی تابع sleep انجام شود. زمان‌بندی براساس توان به کمک نخ idle انجام می‌شود که ممکن است در استفاده از پردازشگر، الگوهایی را کشف کند و پارامترهای هسته را تنظیم نماید تا در مصرف انرژی صرفه‌جویی گردد. فرمان Dev-mode() که به‌وسیله سیستم‌عامل انجام می‌شود می‌تواند برای بیکار نمودن و یا خاموش کردن یک قسمت، براساس شرایط برنامه استفاده شود تا به این ترتیب در مصرف انرژی صرفه‌جویی شود. این فرمان باید مجددا تنظیم شود، تا دوباره از آن استفاده گردد.
- **سیستم‌عامل سنأ اِس:** این سیستم‌عامل، مدیریت توان را به‌صورت یک پروتکل لایه برنامه برای مدیریت شبکه حسگر انجام می‌دهد و از الگوریتم مدیریت توان دینامیکی (DPM) استفاده می‌کند که امکان آهنگ توقف را در زمان اجرا مشخص می‌نماید. DPM را می‌توان به‌صورت یک مدل ماشین حالت محدود بیان کرد.
- **سیستم‌عامل آیس/پیروس:** یکی از معیارهای طراحی سیستم‌عامل آیس/پیروس، پشتیبانی از خط فرمان توان پایین است. این کار را در پایین‌ترین لایه ممکن، یعنی سخت‌افزار انجام می‌دهند. سیستم‌عامل آیس، گره حسگر چندین وضعیت صرفه‌جویی در توان را فراهم می‌کند. گیرنده-فرستنده TR101 که در این گره استفاده می‌شود، کم‌مصرف است. وقتی هیچ رخداد خارجی دیگری برای انجام فرآیند وجود نداشته باشد، این گره وارد وضعیت کم‌مصرف می‌شود. رخدادهای بیرونی، گره را در وضعیت کاری نرمال قرار می‌دهند.
- **سیستم‌عامل اِس‌أ اِس:** مدیریت توان در سیستم‌عامل اِس‌أاِس چندان جدی مورد توجه قرار نگرفته است. در نسخه‌های اخیر هسته اِس‌أ اِس، قطعه هسته ۲۴۳۸DS گنجانده شده است تا صریحاً باتری را در گره پایش کند. این قطعه به واسطها امکان می‌دهد تا به ولتاژ جریان باتری و دمای تراشه ۲۴۳۸DS دسترسی داشته باشند.
- **سیستم‌عامل کانتیکی:** این سیستم‌عامل اگر انتزاع‌های مدیریت توان صریح را فراهم نکند، باز هم این قابلیت را دارد که به برنامه‌ریزان امکان دهد که این مکانیزم‌ها را اجرا نمایند. چنین شرطی سبب می‌شود که حالت صف داخلی آن بر برنامه‌ها تحمیل شود. این برنامه‌ها می‌توانند در مورد قطع توان سیستم در موقع عدم وجود رخدادی برای زمان‌بندی تصمیم بگیرند. پردازشگر در واکنش به رخداد خارجی بیدار می‌شود که به‌وسیله یک اداره‌کننده بیرونی اداره می‌شود.
- **سیستم‌عامل نانوآرکی:** سیستم‌عامل نانوآرکی انواع واسط برنامه‌ریزی کاربردی را فراهم می‌کند. در جدول (۳) واسطهای مدیریت توان آمده است.

جدول ۳- واسطهای سیستم‌عامل نانوآرکی [۴]

API	Functionality
query_energy()	Query residual battery energy
set_energy_mode()	Set energy savings mode (future)
get_energy_mode()	Get energy savings mode (future)
tx_power_set()	Change radio transmitter power
powerdown()	Power the system down for t seconds

۳-۷- مدیریت حافظه

سیستم‌عامل نانوآرکی تنها از مدیریت‌حافظه استاتیکی پشتیبانی می‌کند و از مدیریت حافظه دینامیکی پشتیبانی نمی‌کند. در اینجا، هم سیستم‌عامل و هم برنامه‌ها در یک فضای آدرس منفرد قرار دارند و باید توجه داشت که نانوآرکی کمکی به حفاظت از فضاهای آدرس در فرایند و سیستم‌عامل نمی‌کند.

سیستم‌عامل تینی برای مدیریت حافظه، از مدیریت حافظه استاتیک استفاده می‌کند؛ آن همچنین از حافظه حافظت می‌کند.

1- Priority Scheduling
2- Rate Monotonic Scheduling Algorithm
3- HPLPowerManagement.Enable()
4- StdControl.Init()
5- mos-enable- power- mgt()

سیستم‌عامل‌های کانتیکی و مانتیس از مدیریت حافظه دینامیکی استفاده می‌کنند و از حافظه حافظت نمی‌کنند.

۴- سخت‌افزارهای قابل پشتیبانی

گره‌های سخت‌افزار براساس شرایط کاری و جایی که لازم است استفاده شوند، طراحی می‌شوند. همان‌طور که ذکر شد سیستم‌عامل‌ها دارای خصوصیات و ویژگی‌های مختلفی هستند. بنابراین هر گره براساس آن شرایط، دسته‌ای از سیستم‌عامل‌ها را پشتیبانی می‌کند که با شرایط محیط سازگار باشند. جدول (۴) پشتیبانی سخت‌افزار از سیستم‌عامل‌های موجود را نشان می‌دهد.

جدول ۴- مقایسه سیستم‌عامل‌های شبکه حسگر از لحاظ پشتیبانی سخت‌افزار [۴]

TinyOS	SOS	Contiki	MantisOS
Telos	Cricket	avr MCU	Mica2
Mica2Dot	imote2	MSP430 MCU	MicaZ
Mica2	Mica2	x86	Telos
Mica	Micaz	6502	Mantis nymph
TMote Sky	tmote		
Eyes	XYZ		
MicaZ	Protosb		
iMote	avrora		
	cyclops		
	emu		

۵- مقایسه چند سیستم‌عامل شبکه حسگر و تحلیل آن

در این قسمت براساس ویژگی‌هایی که بیان شد، تعدادی از سیستم‌عامل‌ها را در جدول (۵) با هم مقایسه کرده و سپس تحلیلی در چگونگی استفاده از آن‌ها با توجه به نیازهای مورد نظر، ارائه خواهیم داد.

با توجه به مکان و فعالیتی که حسگرها در آن بکارگیری می‌شوند می‌توان از زمان‌بندی اولویت‌دار استفاده کرد. در مراکز نظامی، هدایت موشک، جنگنده، فعالیت‌های هسته‌ای، شیمیایی، مکان‌های حساس و... باید از سیستم‌عامل‌های زمان‌بندی اولویت‌دار استفاده نمود تا با اولویت‌بندی کارها، کارهای با اولویت بالا سریع‌تر انجام شوند. مدیریت حافظه با توجه به کاری که گره انجام می‌دهد انتخاب می‌شود. در فعالیت‌هایی که نیاز به سرعت انجام می‌باشد و سیستم‌عامل نمی‌خواهد که خود را درگیر مدیریت حافظه دینامیک برای استفاده موثر از حافظه نماید، از مدیریت حافظه استاتیک استفاده می‌شود. چندبرنامه‌گی نیز نقش مهمی در سرعت و استفاده موثر از پردازنده دارد. در مراکز و فعالیت‌های نظامی، به رویدادها باید سریعا پاسخ داده شود. بنابراین در سیستم‌عاملی که برای فعالیت‌های نظامی به‌کار برده می‌شود باید چندبرنامه‌گی و مدیریت حافظه استاتیک را لحاظ نمود. با توجه به بی‌درنگ بودن فعالیت‌های حساس نظامی، استفاده از سیستم‌عاملی که این ویژگی را داشته باشد نیز ضروری است. معماری یکپارچه، نقش مهمی در سرعت اجرا دارد. گره‌های حسگری که در مراکز نظامی به کار برده می‌شوند باید سیستم‌عامل‌های آنها نیز این ویژگی را داشته باشند. سیستم‌عامل نانوآرکی ویژگی‌های بیان شده را دارد؛ بنابراین برای استفاده در مراکز نظامی گزینه خوبی می‌باشد.

جدول ۵- مقایسه تعدادی سیستم‌عامل‌های شبکه حسگر

سیستم عامل	معماری	مدل اجرایی	مدیریت حافظه	پشتیبانی از کاربردهای بی‌درنگ	زمان‌بندی اولویت دار
تینی	یکپارچه	رویدادمحور	مدیریت حافظه استاتیک همراه حافظه‌از حافظه	خیر	خیر
کانتیکی	پیمانه‌ایی	هیبریدی	مدیریت حافظه دینامیک	خیر	خیر
مانتیس	لایه‌ایی	نخ‌محور	مدیریت حافظه دینامیک	خیر	خیر
نانوآرکی	یکپارچه	نخ‌محور	مدیریت حافظه استاتیک	بله	بله
اس‌اُاس	پیمانه‌ایی	رویدادمحور	مدیریت حافظه دینامیک	خیر	بله
سن‌اُاس	پیمانه‌ایی	ماشین حالت محدود	مدیریت حافظه دینامیک	خیر	خیر
آیس	پیمانه‌ایی	رویدادمحور	مدیریت حافظه استاتیک	خیر	بله
پیروس	پیمانه‌ایی	رویدادمحور	مدیریت حافظه استاتیک	خیر	بله
کُورموس	پیمانه‌ایی	رویدادمحور	مدیریت حافظه دینامیک	بله	خیر

۶- نتیجه‌گیری

طراحی و انتخاب سیستم‌عامل شبکه حسگر با طراحی سیستم‌عامل‌های معمولی متفاوت است؛ زیرا در این نوع طراحی باید ویژگی‌های خاص و قابل توجه‌ای مانند محدودیت‌های اندازهٔ فیزیکی، منبع انرژی، قدرت پردازش و ظرفیت حافظه را در نظر گرفت. یکی از کاربردهای شبکه حسگر، مباحث نظامی است. امروزه سپاه نیازمند به‌کارگیری مؤثر این دانش در کنترل دریا و مرزهای زمینی، تشخیص آلودگی شیمیایی و میکروبی محیط و راهبری نیرو و تجهیزات در منطقه نظامی می‌باشد. در سایت‌های موشکی، شناورها، زیردریایی‌ها و هواپیماهای بدون سرنشین استفاده از حسگرها یک امر بدیهی است. تصمیم پرتاب موشک از بهترین سایت موشکی احتمال برخورد، تصمیم عملیات درست در هواپیماهای بدون سرنشین و پایش محیط برای زیردریایی‌ها با استفاده از این دانش به‌راحتی انجام می‌شود. هر یک از این کاربردها ویژگی‌ها و خصوصیات خاصی از نظر مکان، فعالیت و اقدام دارند.

یکی از سیستم‌عامل‌هایی که می‌توان آن را در امور نظامی، حساس و فعالیت‌های بی‌درنگ مورد استفاده قرار داد، سیستم‌عامل نانوآرکی است. این سیستم‌عامل، یک سیستم‌عامل بی‌درنگ چندوظیفه‌ای که دارای زمان‌بندی اولویت‌دار است، در جاهای خطرناک قابل استفاده می‌باشد و از ارتباطات چندگام در شبکه پشتیبانی می‌کند.

بنابراین در فعالیت‌های نظامی با توجه به مکانی که حسگرها به‌کار می‌روند و وظایفی که آنها باید انجام دهند، باید سیستم‌عامل خاص را براساس آن ویژگی‌ها انتخاب نمود.

ویژگی‌های امنیتی سیستم‌عامل، یکی دیگر از عوامل مهم در انتخاب سیستم‌عامل شبکه حسگر می‌باشد که در مقاله آتی مورد بررسی قرار خواهد گرفت.

مراجع

1. I. F. Akyildiz, et al., "Wireless sensor networks: a survey," Computer Networks Vol. 38, pp. 393-422, (**2002**).
2. I. F. Akyildiz and I. H. Kasimoglu "Wireless sensor and actor networks," Ad Hoc Networks Vol. 2, pp. 351-367, (**2004**).
3. E. Barjinder Singh Kaler and E. Manpreet Kaur Kaler "Challenges in Wireless Sensor Networks".
4. A. M. Reddy, et al., "Operating Systems for Wireless Sensor Networks: A Survey Technical Report," (**2007**).
5. M. O. Farooq and T. Kunz, "Operating Systems for Wireless Sensor Networks: A Survey," Sensors (Basel), Vol. 11, pp. 5900-5930, (**2011**).
6. C. Lynch and F. O' Reilly, "Processor Choice For Wireless Sensor Networks," Workshop on Real-World Wireless Sensor Networks (REALWSN'۰5), Stockholm, Sweden, pp. 1-5, June (**2005**).
7. T Stathopoulos, et al., " A remote code update mechanism for wireless sensor networks. Technical Report CENS-TR-3۰, University of California, Los Angeles, Center for Embedded Networked Computing," November (**2003**).
8. J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," presented at the Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, (**2004**).
9. L. Sha, et al., " Priority inheritance protocols: An approach to real-time synchronization," IEEE Transactions on Computers, Vol. 39, pp. 1175-1185, (**1990**).
10. R. Barr, et al., "On the need for system-level support for ad hoc and sensor networks," SIGOPS Oper. Syst. Rev., Vol. 36, pp. 1-5, (**2002**).
11. S Bhatti, et al., "MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms," Mob. Netw. Appl., Vol. 1۰, pp. 563-579, (**2005**).
12. A. Dunkels, et al., "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," presented at the Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, (**2004**).
13. J. Lifton, et al., "Pushpin Computing System Overview: A Platform for Distributed, Embedded, Ubiquitous Sensor Networks," presented at the Proceedings of the First International Conference on Pervasive Computing, (**2002**).
14. C.-C. Han, et al., "A dynamic operating system for sensor nodes," presented at the Proceedings of the ۳rd international conference on Mobile systems, applications, and services, Seattle, Washington, (**2005**).
15. J. Hill, et al., "System architecture directions for networked sensors," SIGARCH Comput. Archit. News, Vol. 28, pp. 93-104, (**2000**).
16. P. Levis, et al., "TinyOS: An Operating System for Sensor Networks ", Ambient Itelligence In Ambient In Intelligence, pp. 115-148, (**2005**).
17. S. Dulman and P. Havinga, "Operating system fundamentals for the EYES distributed sensor network. I, Utrecht, the Netherlands," in In Proceedings of Progress 2002, Utrecht, the Netherlands, October (**2002**).
18. M.-J. Chen and A. C. Bovik, "Fast structural similarity index algorithm," J. Real-Time Image Process., Vol. 6, pp. 281-287, (**2011**).
19. J. Yannakopoulos and A. Bilas, "COMMON-Sense Net Working draft for MICA2 motes" CEDT, (**2004**).
20. R. Shea, et al., "Motivations Behind SOS," SOS1-2000, University of California Los Angeles, Networked Embedded Systems Lab, Los Angeles (**2004**).

Review of Operating Systems of Sensor Network

A. Naseri[1]

M. Ghayouri Sales[2]

M. Naghavi[3]

Abstract

One of the principles of passive defense is the timely detection of the enemy's intrusion to a country's perimeter. Sensors are used as one of the necessary means of identifying and detecting the enemy's intrusion and attack. All sensors need operating systems to identify and detect and do their tasks in military, sensitive and defense applications. Architecture, implementation model, reprogramming, timing and energy management are the important features of operating systems which are effective in the selection of operating systems for sensor networks in military applications. In this essay, we introduce the features of operating systems of sensor networks. We intend to select the best operating systems for sensors used in military centers. Then we review and categorize some of the operating systems of sensor networks such as MantisOS، TinyOS، Contiki، Nano-RK، SOS، Bertha، CORMOS ، EYES/PEEROS based on these features. In the end we introduce the Nano-RK operating system considering the characteristics of sensitive military activities as an option for these sensors.

Key Words: *Wireless Sensor System, Operating System of Wireless Sensor Network*

1- Imam Hossein University, Master of Science Student in Software (nasseri1355@yahoo.com) - Writer in Charge
2- Imam Hossein University, Assistant Professor and Academic Member of Computer Department
3- Imam Hossein University, Graduate of Computer Department